特性

- 融合了 ARM920T™ ARM® Thumb® 处理器
  - 工作于 180 MHz 时性能高达 200 MIPS，存储器管理单元
  - 16-K 字节的数据缓存， 16-K 字节的指令缓存，写缓冲器
  - 含有调试信道的内部仿真器
  - 中等规模的嵌入式宏单元结构 (仅针对 256 BGA 封装)
- 低功耗：VDDCORE 电流为 30.4 mA 待机模式电流为 3.1 mA
- 附加的嵌入式存储器
  - SRAM 为 16K ； ROM 为 128K
- 外部总线接口 (EBI)
  - 支持 SDRAM，静态存储器，Burst Flash，无缝连接的 CompactFlash®，SmartMedia™ 及 NAND Flash
- 提高性能而使用的系统外设：
  - 增强的时钟发生器与电源管理控制器
  - 两个有双 PLL 的片上振荡器
  - 低速的时钟操作模式与软件功耗优化能力
  - 四个可编程的外部时钟信号
  - 包括周期性中断、看门狗及第二计数器的系统定时器
  - 有报警中断的实时时钟
  - 调试单元、两线 UART 并支持调试信道
  - 有 8 个优先级的高级中断控制器，独立的可屏蔽中断源，伪中断保护
  - 7 个外部中断源及 1 个快速中断源
  - 有 122 个可编程 I/O 口线的四个 32 位 PIO 控制器，各线均有输入变化中断及开漏能力
  - 20 通道的外设数据控制器 (DMA)
- 10/100 Base-T 型以太网卡接口
  - 独立的媒体接口 (MII) 或简化的独立媒体接口 (RMII)
  - 对于接收与发送有集成的 28 字节 FIFO 及专用的 DMA 通道
- USB 2.0 全速 (12 M 比特 / 秒) 主机双端口
  - 双片上收发器 (208 引脚 PQFP 封装中仅为一个)
  - 集成的 FIFO 及专用的 DMA 通道
- USB 2.0 全速 (12 M 比特 / 秒) 器件端口
  - 片上收发器，2-K 字节可配置的集成 FIFO
- 多媒体卡接口 (MCI)
  - 自动协议控制及快速自动数据传输
  - 与 MMC 及 SD 存储器卡兼容，支持两个 SD 存储器
- 3 个同步串行控制器 (SSC)
  - 每个接收器与发送器有独立的时钟及帧同步信号
  - 支持 I^2S 模拟接口，时分复用
  - 32 比特的高速数据流传输能力
- 4 个通用同步 / 异步接收 / 发送器 (USART)
  - 支持 ISO7816 T0/T1 智能卡
  - 硬软件握手
  - 支持 RS485 及高达 115 Kbps 的 IrDA 总线
  - USART1 为全调制解调控制线
- 主机 / 从机串行外设接口 (SPI)
  - 8 ～ 16 位可编程数据长度，可连接 4 个外设
- 两个 3 通道 16 位定时 / 计数器 (TC)
  - 3 个外部时钟输入，每条通道有 2 个多功能 I/O 引脚
  - 双 PWM 产生器，捕获 / 波形模式，上加 / 下减计数能力
- 两线接口 (TWI)
  - 主机模式支持，所有两线 Atmel EEPROM 支持
- 所有数字引脚的 IEEE 1149.1 JTAG 边界扫描
- 电源供应
  - VDDCORE， VDDOSC 及 VDDPLL 电压为：1.65V ～ 1.95V
  - VDDIOP (外设 I/O) 及 VDDIOM (存储器 I/O) 电压为：1.65V ～ 3.6V
- 提供了 208 引脚 PQFP 及 256 球状 BGA 两种封装



基于 ARM920T™ 的微控制器

AT91RM9200

本文是英文数据手册的中文翻译，其目的是方便中国用户的阅读。它无法自动跟随原稿的更新，同时也可能存在翻译上的错误。读者应该以英文原稿为参考以获得更准确的信息。



Rev. 1768B-ATARM–07-Jun-05



说明

AT91RM9200是完全围绕ARM920T ARM Thumb处理器构建的系统。它有丰富的系统与应用外设及标准的接口，从而为低功耗、低成本、高性能的计算机宽范围应用提供一个单片解决方案。

AT91RM9200包括一个高速片上SRAM工作区及一个低等待时间的外部总线接口(EBI)，以完成应用所要求的片外存储器和内部存储器映射外设配置的无缝连接。EBI 有同步 DRAM (SDRAM)、Burst Flash 及静态存储器的控制器，并设计了专用电路以方便与 SmartMedia、CompactFlash 及 NAND Flash 连接。

高级中断控制器 (AIC) 通过多向量 ，中断源优先级划分及缩短中断处理传输时间来提高 ARM920T 处理器的中断处理性能。

外设数据控制器 (PDC) 向所有的串行外设提供 DMA 通道，使其与片内或片外存储器传输数据时不用经过处理器。这就减少了传输连续数据流时处理器的开销。包含双指针的 PDC 控制器极大的简化了 AT91RM9200 的缓冲器链接。

并行 I/O (PIO) 控制器与作为通用数据的 I/O 复用外设输入 / 输出口线，以最大程度上适应器件的配置。每条口线上包含有一个输入变化中断、开漏能力及可编程上拉电阻。

电源管理控制器 (PMC) 通过软件控制有选择的使能 / 禁用处理器及各种外设来使系统的功耗保持最低。它用一个增强的时钟产生器来提供包括慢时钟 (32 kHz) 在内的选定时钟信号，以随时优化功耗与性能。

AT91RM9200 集成了许多标准接口，包括 USB 2.0 全速主机和设备端口及与多数外设和在网络层广泛使用的 10/100 Base-T 以太网媒体访问控制器 (MAC)。此外，它还提供一系列符合工业标准的外设，可在音频、电信、 Flash 卡红外线及智能卡中使用。

为完善性能，AT91RM9200 集成了包括 JTAG-ICE、专门 UART 调试通道 (DBGU) 及嵌入式的实时追踪的一系列的调试功能。这些功能使得开发、调试所有的应用特别是受实时性限制的应用成为可能。

方框图

实箭头 (⟶) 表示由主机到从机

Figure 1. AT91RM9200 方框图

TST0-TST1
NRST
Reset and Test
JTAGSEL
TDI
TDO
TMS
TCK
NTRST
JTAG Scan
ICE
ARM920T Core
ETM
Instruction Cache 16K bytes
MMU
Data Cache 16K bytes
PIO
TSYNC
TCLK
TPS0 - TPS2
TPK0 - TPK15
BMS
FIQ
IRQ0-IRQ6
PCK0-PCK3
AIC
Fast SRAM 16K bytes
Address Decoder
Abort Status
Misalignment Detector
Bus Arbiter
Memory Controller
EBI
CompactFlash NAND Flash SmartMedia
SDRAM Controller
Burst Flash Controller
Static Memory Controller
PLLRCB
PLLRCA
XIN
XOUT
PLLB
PLLA
OSC
PMC
System Timer
Fast ROM 128K bytes
Peripheral Bridge
Peripheral DMA Controller
XIN32
XOUT32
OSC
RTC
DRXD
DTXD
DBGU
PDC
PIOA/PIOB/PIOC/PIOD Controller
D0-D15
A0/NBS0
A1/NBS2/NWR2
A2-A15/A18-A22
A16/BA0
A17/BA1
NCS0/BFCS
NCS1/SDCS
NCS2
NCS3/SMCS
NRD/NOE/CFOE
NWR0/NWE/CFWE
NWR1/NBS1/CFIOR
NWR3/NBS3/CFIOW
SDCK
SDCKE
RAS-CAS
SDWE
SDA10
BFRDY/SMOE
BFCK
BFAVD
BFBAA/SMWE
BFOE
BFWE
A23-A24
A25/CFRNW
NWAIT
NCS4/CFCS
NCS5/CFCE1
NCS6/CFCE2
NCS7
D16-D31
DMA
FIFO
USB Host
Transceiver
HDMA
HDPA
HDMB
HDPB
DDM
DDP
Transceiver
FIFO
USB Device
DMA
FIFO
Ethernet MAC 10/100
ETXCK-ERXCK-EREFCK
ETXEN-ETXER
ECRS-ECOL
ERXER-ERXDV
ERX0-ERX3
ETX0-ETX3
EMDC
EMDIO
EF100
MCCK
MCCDA
MCDA0-MCDA3
MCCDB
MCDB0-MCDB3
MCI
PDC
APB
RXD0
TXD0
SCK0
RTS0
CTS0
USART0
PDC
SSC0
PDC
TF0
TK0
TD0
RD0
RK0
RF0
RXD1
TXD1
SCK1
RTS1
CTS1
DSR1
DTR1
DCD1
RI1
USART1
PDC
SSC1
PDC
TF1
TK1
TD1
RD1
RK1
RF1
RXD2
TXD2
SCK2
RTS2
CTS2
USART2
PDC
SSC2
PDC
TF2
TK2
TD2
RD2
RK2
RF2
RXD3
TXD3
SCK3
RTS3
CTS3
USART3
PDC
Timer Counter
TC0
TC1
TC2
TCLK0
TCLK1
TCLK2
TIOA0
TIOB0
TIOA1
TIOB1
TIOA2
TIOB2
NPCS0
NPCS1
NPCS2
NPCS3
MISO
MOSI
SPCK
SPI
PDC
Timer Counter
TC3
TC4
TC5
TCLK3
TCLK4
TCLK5
TIOA3
TIOB3
TIOA4
TIOB4
TIOA5
TIOB5
TWD
TWCK
TWI

主要特性

本节介绍各个模块的主要特性。

ARM920T 处理器

- ARM9TDMI™ 基于 ARM® v4T 架构
- 两套指令集
  - 32 位高性能 ARM® 指令集
  - 16 位高代码密度 Thumb® 指令集
- 5 级流水线结构：
  - 取指令 (F)
  - 指令译码 (D)
  - 执行 (E)
  - 数据存储器 (M)
  - 写寄存器 (W)
- 16-K 字节数据缓存，16-K 字节指令缓存
  - 虚拟地址的 64 路相关缓存
  - 每线 8 字
  - 正向及反向写操作
  - 伪随机或循环置换
  - 低功耗 CAM RAM 设备
- 写缓冲器
  - 16 字的数据缓冲器
  - 4 地址的地址缓冲器
  - 软件控制消耗
- 标准的 ARMv4 存储器管理单元 (MMU)
  - 区域访问许可
  - 允许以 1/4 页面大小对页面进行访问
  - 16 个嵌入域
  - 64 个输入指令 TLB 及 64 个输入数据 TLB
- 8 位、 16 位、 32 位的指令总线与数据总线

调试与测试

- 集成了嵌入式内部电路仿真器
- 调试单元
  - 两引脚的 UART
  - 调试信道
  - 芯片 ID 寄存器
- 嵌入式追踪宏单元：ETM9 Rev2a
  - 中级实现
  - 半速时钟模式
  - 四对地址比较器
  - 两个数据比较器
  - 八个存储器映射解码器输入
  - 两个计数器
  - 一个序列发生器
  - 一个 18 字节的 FIFO
- 数字引脚通过 IEEE1149.1 JTAG 边界扫描

引导程序

- 引导程序默认存储在 ROM 中
- 由外部存储器载入内部 SRAM 中运行
- 下载代码大小由内部 SRAM 大小决定
- 自动检测有效的应用程序
- 引导载入支持多数非易失性存储器
  - 连接在 SPI NPCS0 上的 SPI DataFlash®
  - 两线 EEPROM
  - 若器件集成了 EBI，则在 NCS0 上提供 8 位并行存储器
- 提供支持多种通信介质的引导上传器 (Boot Uploader) 以防外部 NVM 上未检测到有效程序
- DBGU (XModem 协议) 上串行通信
- USB 器件端口 (DFU 协议)

嵌入式软件服务

- ATPCS 适用
- AINSI/ISO 标准 C 适用
- 在 ARM/Thumb 交互工作中编译
- ROM 进入服务
- 提供 Tempo、 Xmodem 及 DataFlash 服务
- CRC 及正弦表

复位控制器

- 提供两条复位输入线 (NRST 与 NTRST)：
- 初始化用户接口寄存器 (各个外设通过用户接口来定义) 且：
  - 在 bootup 时对信号采样
  - 强迫处理器读取零地址空间的下条指令
- 初始化嵌入式 ICE TAP 控制器

存储控制器

- 可编程的对四主机总线仲裁处理
  - 内部总线由 ARM920T、 PDC、 USB 主机端口与以太网 MAC 主机共享
  - 每个主机优先级在 0 ～ 7 之间分配
- 地址解码器提供如下选择：
  - 八个 256-M 字节外部存储器区域
  - 四个 1-M 字节内部存储器区域
  - 一个 256-M 字节嵌入式外设区域
- 引导模式选项：
  - 非易失性引导存储器可为片内或片外的
  - 由 BMS 引脚在复位时的采样值选定
- 异常中断状态寄存器
  - 保存所有引起发生异常中断的源、类型及访问参数
- 检测器失调
  - 对所有数据访问进行校准检测
  - 失调时产生中止
- 重新映射命令
  - 对内部 SRAM 提供重新映射以代替引导 NVM

外部总线接口

- 集成了三个外部存储控制器：



  - 静态存储控制器
  - SDRAM 控制器
  - Burst Flash 控制器
- 额外的支持 SmartMedia™ 及 CompactFlash™ 的逻辑
- 优化外部总线：
  - 16 或 32 位数据总线
  - 26 位地址总线，可对 64-M 字节空间进行寻址
  - 8 个片选信号，每个对应八个存储区域中的一个
  - 优化引脚复用以减少外部存储器等待时间
- 可配置的片选：
  - NCS0 上 Burst Flash 控制器或静态存储控制器
  - NCS1 上 SDRAM 控制器或静态存储控制器
  - NCS3 上静态存储控制器，可选 SmartMedia
  - NCS4 - NCS6 上静态存储控制器，可选 CompactFlash
  - NCS7 上静态存储控制器

静态存储控制器

- 外部存储器有 512-M 字节地址空间
- 8 个片选口线
- 8 位或 16 位数据总线
- 引导存储器的重新映射
- 支持多路访问模式
  - 字节写或字节选择线
  - 每个存储器区有两个不同的读协议
- 多设备适应性
  - LCD 模块适应
  - 可编程启动定时读 / 写
  - 可编程保持定时读 / 写
- 多等待状态管理
  - 可编程等待状态产生
  - 外部等待请求
  - 可编程数据浮动时间

SDRAM 控制器

- 支持多种配置
  - 2K、 4K、 8K 行地址存储部分
  - 两个或四个内部 SDRAM 区
  - 16 位或 32 位数据路径的 SDRAM
- 编程性能
  - 字、半字、字节访问
  - 到达存储器边界时自动分页
  - 多组 Ping-pong 访问
  - 软件确定定时参数
  - 自动更新操作，可编程更新速率
- 节能能力

  - 支持自更新与低功耗模式
- 错误检测
  - 更新错误中断
- 软件上电初始化 SDRAM
- 等待时间为两个时钟 (CAS 等待时间为一个时钟，不支持三个时钟)
- 未使用自动预充电命令

Burst Flash 控制器

- 支持多路访问模式
  - 异步或 Burst 模式字节，半字或字访问
  - 异步模式半字写访问
- 可适应不同速率的器件
  - 可编程 Burst Flash 时钟速率
  - 可编程数据访问时间
  - 可编程输出使能后的等待时间
- 可适应不同的访问协议及总线接口
  - 两个 Burst 读协议：时钟控制地址提前或信号控制地址提前
  - 多路或独立的地址与数据总线
  - 支持连续 Burst 与页模式访问

外设数据控制器

- 通过诸如 DBGU、 USART、 SSC、 SPI 及 MCI 等与外设进行数据传输
- 二十路通道
- 由存储器到外设传输需一个主机时钟周期
- 由外设到存储器传输需两个主机时钟周期

增强的中断控制器

- 控制 ARM® 处理器中断线 (nIRQ 与 nFIQ)
- 32 个可独立屏蔽的中断源向量
  - 中断源 0 为快速中断输入 (FIQ)
  - 中断源 1 为系统外设 (ST、 RTC、 PMC、 DBGU…)
  - 中断源 2 到中断源 31 控制 30 个嵌入式外设中断或外部中断
  - 可编程的边沿触发或电平敏感内部中断
  - 可编程的正 / 负边沿触发或高 / 低电平敏感外部源
- 8 级优先权控制器
  - 驱动处理器正常中断
  - 处理 1 ~ 31 个中断源的优先级
  - 高优先级中断可打断低优先级中断的执行
- 定向
  - 优化中断服务程序分支与执行
  - 每个中断源有一个 32 位向量寄存器
  - 中断向量寄存器读当前相应的中断向量
- 保护模式
  - 禁止自动操作可简化调试
- 快速强制
  - 允许通过处理器快速中断将正常中断源重定向
- 通用中断屏蔽



– 提供在不触发中断的情况下处理器与事件同步

电源管理控制器

- 优化整个系统功耗
- 嵌入与控制：
  – 一个主振荡器与一个慢时钟振荡器 (32.768Hz)
  – 两个锁相环 (PLL) 及分频器
  – 时钟预分频
- 提供：
  – 处理器时钟 PCK
  – 主机时钟 MCK
  – USB 时钟 UHPCK 及 UDPCK 分别对应 USB 主机端口与 USB 器件端口
  – USB 器件延迟情况下可编程将 PLL 自动关闭
  – 30 个外设时钟
  – 四个可编程时钟输出：PCK0 ~ PCK3
- 四种工作模式：
  – 正常模式、空闲模式、慢时钟模式及待机模式

系统定时器

- 一个周期计时器，16 位可编程计数器
- 一个看门狗定时器，16 位可编程计数器
- 一个实时计时器，20 位自主运行计数器
- 事件中断

实时时钟

- 低功耗
- 全异步设计
- 万年历
- 可编程周期中断
- 报警与更新同步下载
- 报警控制与定时 / 日历数据更新

调试单元

- 方便 Atmel ARM® 系统调试的系统外设
- 有四个功能
  – 两引脚 UART
  – 支持调试信道 (DCC)
  – 芯片 ID 寄存器
  – 防止 ICE 访问
- 两引脚 UART
  – 执行特征与标准 Atmel USART 完全兼容
  – 具有通用可编程波特率产生器的独立收发器
  – 奇数、偶数、标志或空间奇偶发生器
  – 奇偶、帧及超速错误检测
  – 自动回复、本地回环及远程回环通道模式
  – 中断产生器
  – 支持与接收器与发送器连接的两个 PDC 通道
- 调试信道支持

  - 可见来自 ARM 处理器的 COMMRX 与 COMMTX 信号
  - 中断产生器
- 芯片 ID 寄存器
  - 识别器件版本、嵌入式存储器大小及外设组。

PIO 控制器

- 32 个可编程 I/O 口线
- 通过置位 / 清零寄存器可完全编程
- 各 I/O 口线复用两个外设功能
- 各 I/O 口线 (无论配置为外设还是作为通用功能 I/O 使用)
  - 输入变化中断
  - 毛刺滤波器
  - 多驱动选择使能开漏驱动
  - 可编程 I/O 口线上拉
  - 引脚数据状态寄存器，随时提供引脚电平
- 同步输出，在单写操作中可对几个 I/O 口线进行置位与清零

USB 主机端口

- 开放的 HCI Rev 1.0 标准适用
- USB V2.0 全速与低速标准适用
- 支持低速 1.5 Mbps 与全速 12 Mbps USB 器件
- 主集线器集成两个下游 USB 端口
- 两个内置 USB 收发器
- 支持电源管理
- 作为存储控制器的一个主机操作

USB 器件端口

- USB V2.0 适用， 12 M 比特 / 秒
- 内置 USB V2.0 全速收发器
- 为终点内置双端口 RAM
- 延迟 / 恢复逻辑
- 同步与大量端点的 Ping-pong 模式 (两个存储器组)
- 6 个通用功能端点
  - 端点 0，端点 3：8 字节，无 ping-pong 模式
  - 端点 1，端点 2：64 字节， ping-pong 模式
  - 端点 4，端点 5：256 字节， ping-pong 模式

以太网 MAC

- 与 IEEE 802.3 标准兼容
- 每秒 10 ~ 100 M 比特的数据吞吐能力
- 全双工或半双工操作
- 与物理层接口为 MII 或 RMII
- 寄存器接口可由地址、状态与控制寄存器使用
- DMA 接口，作为存储控制器的一个主机工作
- 信号接收与发送结束产生中断
- 28 字节传输与 28 字节接收 FIFO
- 对传输帧自动填充并产生 CRC
- 地址逻辑校验以识别四个 48 位地址



- 当所有的有效帧拷贝到存储器中时支持混合模式
- 支持物理层管理，通过 MDIO 接口控制报警与更新定时 / 日历数据

串行外设接口

- 支持与串行外设通信
  - 外部解码器有四个片选位，最多支持与 15 个外设通信
  - 串行存储器，如 DataFlash 及三线 EEPROM
  - 串行外设，如 ADC、 DAC、 LCD 控制器、 CAN 控制器与传感器
  - 外部协处理器
- 主机或从机外设总线接口
  - 每个片选 8 位到 16 位可编程数据长度
  - 每个片选可编程相位与极性
  - 每个片选有在连续传输和时钟与数据间可编程传输延时
  - 连续传输间可编程延时
  - 选择模式故障检测
- 通过连接 PDC 通道优化数据传输
  - 收发各一个通道
  - 支持相邻缓冲

两线接口

- 与标准两线串行存储器兼容
- 从机地址为 1、 2、 3 个字节
- 连续读 / 写操作

USART

- 可编程波特率产生器
- 5 ~ 9 位的全双工同步或异步串行通信
  - 异步模式下 1、 1.5 或 2 个停止位或同步模式下 1 或 2 个停止位
  - 奇偶校验位产生与错误检测
  - 帧错误检测，超速错误检测
  - MSB 或 LSB 在先
  - 可选断点产生与检测
  - 8 或 16 的过采样接收频率
  - 可选硬件握手 RTS-CTS
  - 可选调制解调信号管理 DTR-DSR-DCD-RI
  - 接收器停止与发送器时间防护
  - 可选的地址产生与检测的 Multi-drop 模式
- 有驱动器控制信号的 RS485
- ISO7816， T = 0 或 T = 1 协议与智能卡的接口连接
  - NACK 处理，有循环与迭代限制的错误计数器
- IrDA 调制与解调
  - 通信速率达到 115.2 Kbps
- 测试模式
  - 远程回送、本地回送及自动回复
- 两个外设数据控制器通道连接 (PDC)
  - 不通过处理器的缓冲器传输

串行同步控制器

- 在音频与电信应用中使用串行同步通信链接
- 包含一个独立的接收器和发送器以及通用时钟分频器
- 与两个 PDC 通道 (DMA 访问) 连接以降低处理器开销
- 提供一个可配置的帧同步与数据长度
- 接收器与发送器可编程启动帧同步信号的自动检测方式或不同事件检测方式。
- 接收器与发送器包括一个数据信号、一个时钟信号及一个帧同步信号

定时 / 计数器

- 三个 16 位定时器计数器通道
- 功能包括：
  – 频率测量
  – 事件计数
  – 间隔测量
  – 脉冲产生
  – 延迟定时
  – 脉宽调制
  – 上加 / 下减能力
- 各个通道用户可配置，内容包括：
  – 三个外部时钟输入
  – 武功内部时钟输入
  – 两个多功能输入 / 输出信号
- 内部中断信号
- 两个作用于三个 TC 通道的全局寄存器

多媒体卡接口

- 与多媒体卡标准 V 2.2 兼容
- 与 SD 存储器卡标准 V1.0 兼容
- 卡的时钟速率是主机时钟的 2 倍分频
- 当未使用时内置的电源管理将时钟速率降低
- 支持两种插槽
  – 一种是多媒体卡总线 (可达 30 个卡)，另一种是 SD 存储器卡
- 支持数据流、块或多块数据的读写
- 与外设数据控制器通道连接
  – 对大量缓冲器传输时最小化处理器干预

AT91RM9200 产品特性

电源

AT91RM9200 有 5 种 类型的电源引脚：

- VDDCORE 引脚。它用于向内核供电，包括处理器、存储器与外设；电压范围：1.65V ~ 1.95V，一般为 1.8V。
- VDDIOM 引脚。它给外部总线接口 I/O 口线供电；电压范围 : 1.65V ~ 3.6V ，一般为 1.8V、3V 或 3.3V。
- VDDIOP 引脚。它给外设 I/O 口线与 USB 收发器供电；电压范围：1.65V ~ 3.6V，一般为 1.8V、 3V 或 3.3V。[1]
- VDDPLL 引脚。它给 PLL 供电；电压范围：1.65V ~ 1.95V，一般为 1.8V nominal.
- VDDOSC 引脚。它给振荡器供电；电压范围：1.65V ~ 1.95V，一般为 1.8V nominal.

Note: 1. 若VDDIOP电压低于3V ，则不能使用USB主机与器件端口。同样 ，这会影响追踪端口的工作。

VDDIOM 与 VDDIOP 电源见 Table 1 on page 14 与 Table 2 on page 16。它们使对存储器的供电与外设的供电有所不同。

除 VDDPLL 与 VDDOSC 外，其他电源引脚均需接地。对于接地引脚，分别为 GNDPLL 与 GNDOSC。

引脚输出

AT91RM9200 有两种封装：

- 208 引脚 PQFP， 31.2 x 31.2 mm，引脚间距 0.5 mm。
- 256 球状 BGA， 15 x 15 mm，球间距 0.8 mm。

256 球状 BGA 封装特性是 208 引脚 PQFP 封装的延伸。下列特性仅对 256 球状 BGA 封装有效:

- 并行 I/O 控制器 D。
- PIO 控制器 D 上 ETM 端口与输出复用。
- 第二个 USB 主机收发器，打开内置 USB 主机集线器功能。



208 引脚 PQFP 封装引脚输出

Table 1. AT91RM9200 208 引脚 PQFP 封装

引脚序号	信号名称	引脚序号	信号名称	引脚序号	信号名称	引脚序号	信号名称
1	PC24	37	VDDPLL	73	PA27	109	TMS
2	PC25	38	PLLRCB	74	PA28	110	NTRST
3	PC26	39	GNDPLL	75	VDDIOP	111	VDDIOP
4	PC27	40	VDDIOP	76	GND	112	GND
5	PC28	41	GND	77	PA29	113	TST0
6	PC29	42	PA0	78	PA30	114	TST1
7	VDDIOM	43	PA1	79	PA31/BMS	115	NRST
8	GND	44	PA2	80	PB0	116	VDDCORE
9	PC30	45	PA3	81	PB1	117	GND
10	PC31	46	PA4	82	PB2	118	PB23
11	PC10	47	PA5	83	PB3	119	PB24
12	PC11	48	PA6	84	PB4	120	PB25
13	PC12	49	PA7	85	PB5	121	PB26
14	PC13	50	PA8	86	PB6	122	PB27
15	PC14	51	PA9	87	PB7	123	PB28
16	PC15	52	PA10	88	PB8	124	PB29
17	PC0	53	PA11	89	PB9	125	HDMA
18	PC1	54	PA12	90	PB10	126	HDPA
19	VDDCORE	55	PA13	91	PB11	127	DDM
20	GND	56	VDDIOP	92	PB12	128	DDP
21	PC2	57	GND	93	VDDIOP	129	VDDIOP
22	PC3	58	PA14	94	GND	130	GND
23	PC4	59	PA15	95	PB13	131	VDDIOM
24	PC5	60	PA16	96	PB14	132	GND
25	PC6	61	PA17	97	PB15	133	A0/NBS0
26	VDDIOM	62	VDDCORE	98	PB16	134	A1/NBS2/NWR2
27	GND	63	GND	99	PB17	135	A2
28	VDDPLL	64	PA18	100	PB18	136	A3
29	PLLRCA	65	PA19	101	PB19	137	A4
30	GNDPLL	66	PA20	102	PB20	138	A5
31	XOUT	67	PA21	103	PB21	139	A6
32	XIN	68	PA22	104	PB22	140	A7
33	VDDOSC	69	PA23	105	JTAGSEL	141	A8
34	GNDOSC	70	PA24	106	TDI	142	A9
35	XOUT32	71	PA25	107	TDO	143	A10
36	XIN32	72	PA26	108	TCK	144	SDA10

Table 1. AT91RM9200 208 引脚 PQFP 封装

引脚序号	信号名称	引脚序号	信号名称	引脚序号	信号名称	引脚序号	信号名称
145	A11	161	PC7	177	CAS	193	D10
146	VDDIOM	162	PC8	178	SDWE	194	D11
147	GND	163	PC9	179	D0	195	D12
148	A12	164	VDDIOM	180	D1	196	D13
149	A13	165	GND	181	D2	197	D14
150	A14	166	NCS0/BFCS	182	D3	198	D15
151	A15	167	NCS1/SDCS	183	VDDIOM	199	VDDIOM
152	VDDCORE	168	NCS2	184	GND	200	GND
153	GND	169	NCS3/SMCS	185	D4	201	PC16
154	A16/BA0	170	NRD/NOE/CFOE	186	D5	202	PC17
155	A17/BA1	171	NWR0/NWE/CFWE	187	D6	203	PC18
156	A18	172	NWR1/NBS1/CFIOR	188	VDDCORE	204	PC19
157	A19	173	NWR3/NBS3/CFIOW	189	GND	205	PC20
158	A20	174	SDCK	190	D7	206	PC21
159	A21	175	SDCKE	191	D8	207	PC22
160	A22	176	RAS	192	D9	208	PC23

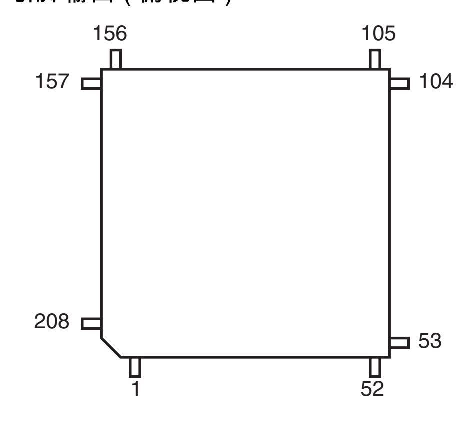
Note: 1. 阴影部分引脚电压由 VDDIOM 提供。

208 引脚 PQFP 封装机械概述

Figure 2 给出 208 引脚 PQFP 封装定位。

详细的机械说明见机械特性部分。

Figure 2. 208 引脚 PQFP 引脚输出 (俯视图)



256 球状 BGA 封装引脚输出

Table 2. AT91RM9200 256 球状 BGA 封装引脚输出

引脚	信号名称	引脚	信号名称	引脚	信号名称	引脚	信号名称
A1	TDI	C3	PD14	E5	TCK	G14	PA1
A2	JTAGSEL	C4	PB22	E6	GND	G15	PA2
A3	PB20	C5	PB19	E7	PB15	G16	PA3
A4	PB17	C6	PD10	E8	GND	G17	XIN32
A5	PD11	C7	PB13	E9	PB7	H1	PD23
A6	PD8	C8	PB12	E10	PB3	H2	PD20
A7	VDDIOP	C9	PB6	E11	PA29	H3	PD22
A8	PB9	C10	PB1	E12	PA26	H4	PD21
A9	PB4	C11	GND	E13	PA25	H5	VDDIOP
A10	PA31/BMS	C12	PA20	E14	PA9	H13	VDDPLLB
A11	VDDIOP	C13	PA18	E15	PA6	H14	VDDIOP
A12	PA23	C14	VDDCORE	E16	PD3	H15	GNDPLLB
A13	PA19	C15	GND	E17	PD0	H16	GND
A14	GND	C16	PA8	F1	PD16	H17	XOUT32
A15	PA14	C17	PD5	F2	GND	J1	PD25
A16	VDDIOP	D1	TST1	F3	PB23	J2	PD27
A17	PA13	D2	VDDIOP	F4	PB25	J3	PD24
B1	TDO	D3	VDDIOP	F5	PB24	J4	PD26
B2	PD13	D4	GND	F6	VDDCORE	J5	PB28
B3	PB18	D5	VDDIOP	F7	PB16	J6	PB29
B4	PB21	D6	PD7	F9	PB11	J12	GND
B5	PD12	D7	PB14	F11	PA30	J13	GNDOSC
B6	PD9	D8	VDDIOP	F12	PA28	J14	VDDOSC
B7	GND	D9	PB8	F13	PA4	J15	VDDPLLA
B8	PB10	D10	PB2	F14	PD2	J16	GNDPLLA
B9	PB5	D11	GND	F15	PD1	J17	XIN
B10	PB0	D12	PA22	F16	PA5	K1	HDPA
B11	VDDIOP	D13	PA21	F17	PLLRCB	K2	DDM
B12	PA24	D14	PA16	G1	PD19	K3	HDMA
B13	PA17	D15	PA10	G2	PD17	K4	VDDIOP
B14	PA15	D16	PD6	G3	GND	K5	DDP
B15	PA11	D17	PD4	G4	PB26	K13	PC5
B16	PA12	E1	NRST	G5	PD18	K14	PC4
B17	PA7	E2	NTRST	G6	PB27	K15	PC6
C1	TMS	E3	GND	G12	PA27	K16	VDDIOM
C2	PD15	E4	TST0	G13	PA0	K17	XOUT

Table 2. AT91RM9200 256 球状 BGA 封装引脚输出

引脚	信号名称
L1	GND
L2	HDPB
L3	HDMB
L4	A6
L5	GND
L6	VDDIOP
L12	PC10
L13	PC15
L14	PC2
L15	PC3
L16	VDDCORE
L17	PLLRCA
M1	VDDIOM
M2	GND
M3	A3
M4	A1/NBS2/NWR2
M5	A10
M6	A2
M7	GND
M9	NCS1/SDCS
M11	D4
M12	GND
M13	PC13
M14	PC1
M15	PC0
M16	GND
M17	PC14
N1	A0/NBS0
N2	A5
N3	A9
N4	A4
N5	A14
N6	SDA10
N7	A8
N8	A21
N9	NRD/NOE/CFOE
N10	RAS
N11	D2
N12	GND
N13	PC28
N14	PC31
N15	PC30
N16	PC11
N17	PC12
P1	A7
P2	A13
P3	A12
P4	VDDIOM
P5	A11
P6	A22
P7	PC9
P8	NWR0/NWE/CFWE
P9	SDCKE
P10	D1
P11	D5
P12	D10
P13	D15
P14	PC26
P15	PC27
P16	VDDIOM
P17	GND
R1	GND
R2	GND
R3	A18
R4	A20
R5	PC8
R6	VDDIOM
R7	NCS3/SMCS
R8	NWR3/NBS3/CFIOW
R9	D0
R10	VDDIOM
R11	D8
R12	D13
R13	PC17
R14	VDDIOM
R15	PC24
R16	PC29
R17	VDDIOM
T1	A15
T2	VDDCORE
T3	A17/BA1
T4	PC7
T5	VDDIOM
T6	NCS2
T7	NWR1/NBS1/CFIOR
T8	SDWE
T9	GND
T10	VDDCORE
T11	D9
T12	D12
T13	GND
T14	PC19
T15	PC21
T16	PC23
T17	PC25
U1	VDDCORE
U2	GND
U3	A16/BA0
U4	A19
U5	GND
U6	NCS0/BFCS
U7	SDCK
U8	CAS
U9	D3
U10	D6
U11	D7
U12	D11
U13	D14
U14	PC16
U15	PC18
U16	PC20
U17	PC22

Note: 1. 阴影部分引脚电压由 VDDIOM 提供。

256球状BGA封装机械概述

Figure 3 on page 18 给出 256 球状 BGA 封装定位。

详细的机械说明见机械特性部分。

Figure 3. 256 球状 BGA 引脚输出 (俯视图)

PIO 口线的外设复用

AT91RM9200 起重要作用的四个 PIO 控制器：

- PIOA 与 PIOB 复用外设 I/O 口线。
- PIOC ,复用数据总线位 16 ~ 31 及几个外部总线接口控制信号。使用 PIOC 引脚增加有效的通用功能 I/O 口线数目但禁止了 32 位存储访问。
- PIOD，只在 256 球状 BGA 封装中有效，复用外设与 ETM 端口输出口线。

每个 PIO 控制器可控制多达 32 个口线。每个口线可配置为两个外设功能 A 或 B 中的一个。下面的表中定义了外设 A 与 B I/O 口线如何复用 PIO 控制器 A、 B、 C、 D。" 功能 " 与 " 注释 " 栏中插入用户自己的注释；它们用来追踪在应用中如何定义引脚。

"复位状态"栏中说明PIO口线复位是在I/O模式还是在外设模式。若为"I/O" ,PIO为上拉使能输入复位 ,器件在 NRST 被检测到后立即保持在静止状态。因此寄存器 PIO_PSR (外设状态寄存器) 中与 PIO 口线的对应位被复位为低。

若 " 复位状态 " 栏中出现了信号名称，PIO 口线将分配给该信号，PIO_PSR 寄存器中的相应位拉高。这是地址线或芯片选择引脚控制存储时的情况，要求当 NRST 上升时驱动引脚。注意，此时上拉电阻也将使能。

详见 Table 3 on page 19、 Table 4 on page 20、 Table 5 on page 21 与 Table 6 on page 22。

PIO 控制器 A 复用÷

Table 3. PIO 控制器 A 复用

PIO 控制器 A				用法	
I/O 口线	**外设 A**	**外设 B**	**复位状态**	**功能**	**注释**
PA0	MISO	PCK3	I/O		
PA1	MOSI	PCK0	I/O		
PA2	SPCK	IRQ4	I/O		
PA3	NPCS0	IRQ5	I/O		
PA4	NPCS1	PCK1	I/O		
PA5	NPCS2	TXD3	I/O		
PA6	NPCS3	RXD3	I/O		
PA7	ETXCK/EREFCK	PCK2	I/O		
PA8	ETXEN	MCCDB	I/O		
PA9	ETX0	MCDB0	I/O		
PA10	ETX1	MCDB1	I/O		
PA11	ECRS/ECRSDV	MCDB2	I/O		
PA12	ERX0	MCDB3	I/O		
PA13	ERX1	TCLK0	I/O		
PA14	ERXER	TCLK1	I/O		
PA15	EMDC	TCLK2	I/O		
PA16	EMDIO	IRQ6	I/O		
PA17	TXD0	TIOA0	I/O		
PA18	RXD0	TIOB0	I/O		
PA19	SCK0	TIOA1	I/O		
PA20	CTS0	TIOB1	I/O		
PA21	RTS0	TIOA2	I/O		
PA22	RXD2	TIOB2	I/O		
PA23	TXD2	IRQ3	I/O		
PA24	SCK2	PCK1	I/O		
PA25	TWD	IRQ2	I/O		
PA26	TWCK	IRQ1	I/O		
PA27	MCCK	TCLK3	I/O		
PA28	MCCDA	TCLK4	I/O		
PA29	MCDA0	TCLK5	I/O		
PA30	DRXD	CTS2	I/O		
PA31	DTXD	RTS2	I/O		



PIO 控制器 B 复用

Table 4. PIO 控制器 B 复用

PIO 控制 B				用法	
I/O 口线	**外设 A**	**外设 B**	**复位状态**	**功能**	**注释**
PB0	TF0	RTS3	I/O		
PB1	TK0	CTS3	I/O		
PB2	TD0	SCK3	I/O		
PB3	RD0	MCDA1	I/O		
PB4	RK0	MCDA2	I/O		
PB5	RF0	MCDA3	I/O		
PB6	TF1	TIOA3	I/O		
PB7	TK1	TIOB3	I/O		
PB8	TD1	TIOA4	I/O		
PB9	RD1	TIOB4	I/O		
PB10	RK1	TIOA5	I/O		
PB11	RF1	TIOB5	I/O		
PB12	TF2	ETX2	I/O		
PB13	TK2	ETX3	I/O		
PB14	TD2	ETXER	I/O		
PB15	RD2	ERX2	I/O		
PB16	RK2	ERX3	I/O		
PB17	RF2	ERXDV	I/O		
PB18	RI1	ECOL	I/O		
PB19	DTR1	ERXCK	I/O		
PB20	TXD1		I/O		
PB21	RXD1		I/O		
PB22	SCK1		I/O		
PB23	DCD1		I/O		
PB24	CTS1		I/O		
PB25	DSR1	EF100	I/O		
PB26	RTS1		I/O		
PB27	PCK0		I/O		
PB28	FIQ		I/O		
PB29	IRQ0		I/O		

PIO 控制器 C 复位

PIO 控制器 C 没有复用，且仅使用外设 A 口线。在 PIO 控制器 C 选择外设 B 无效。

Table 5. PIO 控制器 C 复用

PIO 控制器 C				用法	
I/O 口线	**外设 A**	**外设 B**	**复位状态**	**功能**	**注释**
PC0	BFCK		I/O		
PC1	BFRDY/SMOE		I/O		
PC2	BFAVD		I/O		
PC3	BFBAA/SMWE		I/O		
PC4	BFOE		I/O		
PC5	BFWE		I/O		
PC6	NWAIT		I/O		
PC7	A23		A23		
PC8	A24		A24		
PC9	A25/CFRNW		A25		
PC10	NCS4/CFCS		NCS4		
PC11	NCS5/CFCE1		NCS5		
PC12	NCS6/CFCE2		NCS6		
PC13	NCS7		NCS7		
PC14			I/O		
PC15			I/O		
PC16	D16		I/O		
PC17	D17		I/O		
PC18	D18		I/O		
PC19	D19		I/O		
PC20	D20		I/O		
PC21	D21		I/O		
PC22	D22		I/O		
PC23	D23		I/O		
PC24	D24		I/O		
PC25	D25		I/O		
PC26	D26		I/O		
PC27	D27		I/O		
PC28	D28		I/O		
PC29	D29		I/O		
PC30	D30		I/O		
PC31	D31		I/O		

PIO 控制器 D 复用

PIO 控制器 D 与外设 A 连接的纯输出信号复用，特别是 EMAC RMII 接口与 外设 B 连接的 ETM 端口。

PIO 控制器 D 只在 AT91RM9200 256 球状 BGA 封装中有效。

Table 6. PIO 控制器 D 复用

PIO 控制器 D				用法	
I/O 口线	外设 A	外设 B	复位状态	功能	注释
PD0	ETX0		I/O		
PD1	ETX1		I/O		
PD2	ETX2		I/O		
PD3	ETX3		I/O		
PD4	ETXEN		I/O		
PD5	ETXER		I/O		
PD6	DTXD		I/O		
PD7	PCK0	TSYNC	I/O		
PD8	PCK1	TCLK	I/O		
PD9	PCK2	TPS0	I/O		
PD10	PCK3	TPS1	I/O		
PD11		TPS2	I/O		
PD12		TPK0	I/O		
PD13		TPK1	I/O		
PD14		TPK2	I/O		
PD15	TD0	TPK3	I/O		
PD16	TD1	TPK4	I/O		
PD17	TD2	TPK5	I/O		
PD18	NPCS1	TPK6	I/O		
PD19	NPCS2	TPK7	I/O		
PD20	NPCS3	TPK8	I/O		
PD21	RTS0	TPK9	I/O		
PD22	RTS1	TPK10	I/O		
PD23	RTS2	TPK11	I/O		
PD24	RTS3	TPK12	I/O		
PD25	DTR1	TPK13	I/O		
PD26		TPK14	I/O		
PD27		TPK15	I/O		

引脚名称说明

Table 7 给出按外设分类的引脚名称。

Table 7. 引脚说明表

引脚名称	功能	类型	激活电平	注释
		电源		
VDDIOM	存储器 I/O 口线电源	电源		1.65V ~ 3.6V
VDDIOP	外设 I/O 口线电源	电源		1.65V ~ 3.6V
VDDPLL	振荡器与 PLL 电源	电源		1.65V ~ 1.95V
VDDCORE	内核芯片电源	电源		1.65V ~ 1.95V
VDDOSC	振荡器电源	电源		1.65V ~ 1.95V
GND	地	地		
GNDPLL	PLL 地	地		
GNDOSC	振荡器地	地		
		时钟，振荡器与 PLL		
XIN	主晶体输入	输入		
XOUT	主晶体输出	输出		
XIN32	32KHz 晶体输入	输入		
XOUT32	32KHz 晶体输出	输出		
PLLRCA	PLL A 滤波器	输入		
PLLRCB	PLL B 滤波器	输入		
PCK0 - PCK3	可编程时钟输出	输出		
		ICE 与 JTAG		
TCK	测试时钟	输入		
TDI	测试数据输入	输入		
TDO	测试数据输出	输出		
TMS	测试模式选择	输入		
NTRST	测试复位信号	输入	低	
JTAGSEL	JTAG 选择	输入		
		ETM		
TSYNC	追踪同步信号	输出		
TCLK	追踪时钟	输出		
TPS0 - TPS2	追踪 ARM 流水线状态	输出		
TPK0 - TPK15	追踪分组端口	输出		
		复位 / 测试		
NRST	微控制器复位	输入	低	无片上上拉
TST0 - TST1	测试模式选择	输入		正常工作时尽量拉低
		存储控制器		



Table 7. 引脚说明表

引脚名称	功能	类型	激活电平	注释
BMS	启动模式选择	输入		
		调试单元		
DRXD	调试接收数据	输入		调试接收数据
DTXD	调试发送数据	输出		调试发送数据
		AIC		
IRQ0 - IRQ6	外部中断输入	输入		
FIQ	快速中断输入	输入		
		PIO		
PA0 - PA31	并行 IO 控制器 A	I/O		复位时将输入上拉
PB0 - PB29	并行 IO 控制器 B	I/O		复位时将输入上拉
PC0 - PC31	并行 IO 控制器 C	I/O		复位时将输入上拉
PD0 - PD27	并行 IO 控制器 D	I/O		复位时将输入上拉
		EBI		
D0 - D15	数据总线	I/O		复位时将输入上拉
D16 - D31	数据总线	I/O		复位时将输入上拉
A0 - A25	地址总线	输出		复位时为 0
		SMC		
NCS0 - NCS7	芯片选择口线	输出	低	复位时为 1
NWR0 - NWR3	写信号	输出	低	复位时为 1
NOE	输出使能	输出	低	复位时为 1
NRD	读信号	输出	低	复位时为 1
NUB	选择最高字节	输出	低	复位时为 1
NLB	选择最低字节	输出	低	复位时为 1
NWE	写使能	输出	低	复位时为 1
NBS0 - NBS3	字节屏蔽信号	输出	低	复位时为 1
		EBI 支持 CompactFlash		
CFCE1 - CFCE2	CompactFlash 片使能	输出	低	
CFOE	CompactFlash 输出使能	输出	低	
CFWE	CompactFlash 写使能	输出	低	
CFIOR	CompactFlash IO 读	输出	低	
CFIOW	CompactFlash IO 写	输出	低	
CFRNW	CompactFlash 只读	输出		
CFCS	CompactFlash 片选	输出	低	

Table 7. 引脚说明表

引脚名称	功能	类型	激活电平	注释
	EBI 支持智能媒体			
SMCS	智能媒体片选	输出	低	
SMOE	智能媒体输出使能	输出	低	
SMWE	智能媒体写使能	输出	低	
	SDRAM 控制器			
SDCK	SDRAM 时钟	输出		
SDCKE	SDRAM 时钟使能	输出	高	
SDCS	SDRAM 控制器片选	输出	低	
BA0 - BA1	选择组	输出		
SDWE	SDRAM 写使能	输出	低	
RAS - CAS	行与列信号	输出	低	
SDA10	SDRAM 地址 10 口线	输出		
	Burst Flash 控制器			
BFCK	Burst Flash 时钟	输出		
BFCS	Burst Flash 片选	输出	低	
BFAVD	Burst Flash 地址有效	输出	低	
BFBAA	Burst Flash 地址提前	输出	低	
BFOE	Burst Flash 输出使能	输出	低	
BFRDY	Burst Flash 就绪	输入	高	
BFWE	Burst Flash 写使能	输出	低	
	多媒体卡接口			
MCCK	多媒体卡时钟	输出		
MCCDA	多媒体卡 A 命令	I/O		
MCDA0 - MCDA3	多媒体卡 A 数据	I/O		
MCCDB	多媒体卡 B 命令	I/O		
MCDB0 - MCDB3	多媒体卡 B 数据	I/O		
	USART			
SCK0 - SCK3	串行时钟	I/O		
TXD0 - TXD3	发送数据	输出		
RXD0 - RXD3	接收数据	输入		
RTS0 - RTS3	发送就绪	输出		
CTS0 - CTS3	发送清除	输入		
DSR1	数据就绪	输入		
DTR1	数据终端就绪	输出		
DCD1	数据载波检测	输入		
RI1	环指示器	输入		



Table 7. 引脚说明表

引脚名称	功能	类型	激活电平	注释
USB 器件端口				
DDM	USB 器件端口数据 -	模拟		
DDP	USB 器件端口数据 +	模拟		
USB Host Port				
HDMA	USB 主机端口 A 数据 -	模拟		
HDPA	USB 主机端口 A 数据 +	模拟		
HDMB	USB 主机端口 B 数据 -	模拟		
HDPB	USB 主机端口 B 数据 +	模拟		
以太网 MAC				
EREFCK	参考时钟	输入		仅针对 RMII
ETXCK	发送时钟	输入		仅针对 MII
ERXCK	接收时钟	输入		仅针对 MII
ETXEN	发送使能	输出		
ETX0 - ETX3	发送数据	输出		ETX0 - ETX1，仅针对 RMII
ETXER	发送译码错误	输出		仅针对 MII
ERXDV	接收数据有效	输入		仅针对 MII
ECRSDV	载波检测与数据有效	输入		仅针对 RMII
ERX0 - ERX3	接收数据	输入		ERX0 - ERX1，仅针对 RMII
ERXER	接收错误	输入		
ECRS	载波检测	输入		仅针对 MII
ECOL	冲突检测	输入		仅针对 MII
EMDC	管理数据时钟	输出		
EMDIO	管理数据输入 / 输出	I/O		
EF100	强制 100 Mb/s	输出	高	仅针对 RMII
同步串行控制器				
TD0 - TD2	发送数据	输出		
RD0 - RD2	接收数据	输入		
TK0 - TK2	发送时钟	I/O		
RK0 - RK2	接收时钟	I/O		
TF0 - TF2	发送帧同步	I/O		
RF0 - RF2	接收帧同步	I/O		
定时器 / 计数器				
TCLK0 - TCLK5	外部时钟输入	输入		
TIOA0 - TIOA5	I/O 口线 A	I/O		
TIOB0 - TIOB5	I/O 口线 B	I/O		

Table 7. 引脚说明表

引脚名称	功能	类型	激活电平	注释
		SPI		
MISO	主入从出	I/O		
MOSI	主出从入	I/O		
SPCK	SPI 串行时钟	I/O		
NPCS0	SPI 外设片选 0	I/O	低	
NPCS1 - NPCS3	SPI 外设片选	输出	低	
		两线接口		
TWD	两线串行数据	I/O		
TWCK	两线串行时钟	I/O		



外设标识

AT91RM9200 内置大量外设。Table 8 定义了 AT91RM9200 外设标识。使用增强中断控制器控制及外设时钟的电源管理控制器时需要外设标识。

Table 8. 外设标识

外设 ID	外设助记符	外设名称	外部中断
0	AIC	增强中断控制器	FIQ
1	SYSIRQ		
2	PIOA	并行 I/O 控制器 A	
3	PIOB	并行 I/O 控制器 B	
4	PIOC	并行 I/O 控制器 C	
5	PIOD	并行 I/O 控制器 D	
6	US0	USART 0	
7	US1	USART 1	
8	US2	USART 2	
9	US3	USART 3	
10	MCI	多媒体卡接口	
11	UDP	USB 器件端口	
12	TWI	两线接口	
13	SPI	串行外设接口	
14	SSC0	同步串行控制器 0	
15	SSC1	同步串行控制器 1	
16	SSC2	同步串行控制器 2	
17	TC0	定时器 / 计数器 0	
18	TC1	定时器 / 计数器 1	
19	TC2	定时器 / 计数器 2	
20	TC3	定时器 / 计数器 3	
21	TC4	定时器 / 计数器 4	
22	TC5	定时器 / 计数器 5	
23	UHP	USB 主机端口	
24	EMAC	以太网 MAC	
25	AIC	增强中断控制器	IRQ0
26	AIC	增强中断控制器	IRQ1
27	AIC	增强中断控制器	IRQ2
28	AIC	增强中断控制器	IRQ3
29	AIC	增强中断控制器	IRQ4
30	AIC	增强中断控制器	IRQ5
31	AIC	增强中断控制器	IRQ6

系统中断

系统中断来自中断信号的线或：

- 存储控制器
- 调试单元
- 系统定时器
- 实时时钟
- 电源管理控制器

这些外设时钟不可控，外设 ID 1 仅能在增强中断控制器中使用。

外部中断

所有的外部中断信号，即快速中断信号 FIQ 或中断信号 IRQ0 到 IRQ6，使用一个专门的外设 ID。但时钟控制与这些外设 ID 无关。



产品存储器映射

第一级地址译码由存储控制器执行，即由具有附加功能的高级系统总线 (ASB) 执行。

译码将 4G 的地址空间分为 16 个 256M 字节的区域。区域 1 ～ 8 对应 EBI，和外部片选 NC0 ～ NCS7相联系。区域0为内部存储器地址，第二级译码提供1M字节内部存储空间。区域15为外设地址，且提供对高级外设总线 (APB) 的访问。

其它区域未使用，使用它们进行访问时将向发出访问请求的主机发出异常中断。

外部存储器映射

Figure 4. 外部存储器映射

Size	Start Address	End Address	Region	Controller	Note
256M Bytes	0x0000 0000	0x0FFF FFFF	Internal Memories		
256M Bytes	0x1000 0000	0x1FFF FFFF	Chip Select 0	SMC or BFC	
256M Bytes	0x2000 0000	0x2FFF FFFF	Chip Select 1	SMC or SDRAMC	
256M Bytes	0x3000 0000	0x3FFF FFFF	Chip Select 2	SMC	
256M Bytes	0x4000 0000	0x4FFF FFFF	Chip Select 3	SMC	NAND Flash/SmartMedia
256M Bytes	0x5000 0000	0x5FFF FFFF	Chip Select 4	SMC	CompactFlash
256M Bytes	0x6000 0000	0x6FFF FFFF	Chip Select 5	SMC	CompactFlash
256M Bytes	0x7000 0000	0x7FFF FFFF	Chip Select 6	SMC	CompactFlash
256M Bytes	0x8000 0000	0x8FFF FFFF	Chip Select 7	SMC	
6 x 256M Bytes 1,536 bytes	0x9000 0000	0xEFFF FFFF	Undefined (Abort)		
256M Bytes	0xF000 0000	0xFFFF FFFF	Peripherals		

内部存储器映射

内部 RAM

AT91RM9200 集成了高速，16-K 字节的内部 SRAM。复位后到重新映射命令执行前，只可访问 SRAM 中 0x20 0000 的地址空间。重新映射后， SRAM 在地址 0x0 同样有效。

内部 ROM

AT91RM9200集成了一个128-K字节的内部ROM。任何时候，ROM均被映射到地址0x10 0000。若复位时 BMS 为高，则在复位后到重新映射命令执行前，可访问地址 0x0。

USB 主机端口

AT91RM9200集成了一个USB主机端口开放主机控制器接口(OHCI)。ASB可直接访问该接口寄存器，且同标准内部存储器一样映射到地址 0x30 0000。

Figure 5. 内部存储器映射

Address	Area	Size
0x0000 0000 – 0x000F FFFF	Internal Memory Area 0	1 MBytes
0x0010 0000 – 0x001F FFFF	Internal Memory Area 1 Internal ROM	1 MBytes
0x0020 0000 – 0x002F FFFF	Internal Memory Area 2 Internal SRAM	1 MBytes
0x0030 0000 – 0x003F FFFF	Internal Memory Area 3 USB Host Port	1 MBytes
0x0040 0000 – 0x0FFF FFFF	Undefined Area (Abort)	252M bytes

256Mbytes (0x0000 0000 – 0x0FFF FFFF)

外设映射

系统外设映射

系统外设映射到地址空间的头 4K 字节中，地址范围为：0xFFFF F000 ~ 0xFFFF FFFF。每个外设有 256 或 512 个字节。

Figure 6. 系统外设映射

Start Address	End Address	Peripheral	Peripheral Name	Size
0xFFFF F000	0xFFFF F1FF	AIC	Advanced Interrupt Controller	512 bytes/128 registers
0xFFFF F200	0xFFFF F3FF	DBGU	Debug Unit	512 bytes/128 registers
0xFFFF F400	0xFFFF F5FF	PIOA	PIO Controller A	512 bytes/128 registers
0xFFFF F600	0xFFFF F7FF	PIOB	PIO Controller B	512 bytes/128 registers
0xFFFF F800	0xFFFF F9FF	PIOC	PIO Controller C	512 bytes/128 registers
0xFFFF FA00	0xFFFF FBFF	PIOD	PIO Controller D	512 bytes/128 registers
0xFFFF FC00	0xFFFF FCFF	PMC	Power Management Controller	256 bytes/64 registers
0xFFFF FD00	0xFFFF FDFF	ST	System Timer	256 bytes/64 registers
0xFFFF FE00	0xFFFF FEFF	RTC	Real-time Clock	256 bytes/64 registers
0xFFFF FF00	0xFFFF FFFF	MC	Memory Controller	256 bytes/64 registers

用户外设映射

用户外设映射到地址空间的前 256M 字节中，地址范围：0xFFFA 0000 ~ 0xFFFE 3FFF。每个外设地址空间为 16-K 字节。

Figure 7. 用户外设映射

Start Address	End Address	Peripheral	Peripheral Name	Size
0xF000 0000	0xFFF9 FFFF	Reserved		
0xFFFA 0000	0xFFFA 3FFF	TC0, TC1, TC2	Timer/Counter 0, 1 and 2	16K Bytes
0xFFFA 4000	0xFFFA 7FFF	TC3, TC4, TC5	Timer/Counter 3, 4 and 5	16K Bytes
0xFFFA 8000	0xFFFA FFFF	Reserved		
0xFFFB 0000	0xFFFB 3FFF	UDP	USB Device Port	16K Bytes
0xFFFB 4000	0xFFFB 7FFF	MCI	Multimedia Card Interface	16K Bytes
0xFFFB 8000	0xFFFB BFFF	TWI	Two-Wire Interface	16K Bytes
0xFFFB C000	0xFFFB FFFF	EMAC	Ethernet MAC	16K Bytes
0xFFFC 0000	0xFFFC 3FFF	USART0	Universal Synchronous Asynchronous Receiver Transmitter 0	16K Bytes
0xFFFC 4000	0xFFFC 7FFF	USART1	Universal Synchronous Asynchronous Receiver Transmitter 1	16K Bytes
0xFFFC 8000	0xFFFC BFFF	USART2	Universal Synchronous Asynchronous Receiver Transmitter 2	16K Bytes
0xFFFC C000	0xFFFC FFFF	USART3	Universal Synchronous Asynchronous Receiver Transmitter 3	16K Bytes
0xFFFD 0000	0xFFFD 3FFF	SSC0	Serial Synchronous Controller 0	16K Bytes
0xFFFD 4000	0xFFFD 7FFF	SSC1	Serial Synchronous Controller 1	16K Bytes
0xFFFD 8000	0xFFFD BFFF	SSC2	Serial Synchronous Controller 2	16K Bytes
0xFFFD C000	0xFFFD FFFF	Reserved		
0xFFFE 0000	0xFFFE 3FFF	SPI	Serial Peripheral Interface	16K Bytes
0xFFFE 4000	0xFFFE FFFF	Reserved		

外设执行

USART

USART 可管理调制解调信号 DTR、 DSR、 DCD 与 RI 详见 “Modem Mode” on page 420。

AT91RM9200 中，只有 USART1 执行这些信号。

USART0、USART2 与 USART3 不执行调制解调信号。这些 USART 中只执行用于其他性能的 RTS 与 CTS (RTS0 与 CTS0 ； RTS2 与 CTS2 ； RTS3 与 CTS3) 信号。

因此，在调制解调模式下对 USART0、 USART2 或 USART3 编程，其结果无法预测。在这些 USART 中，与调制解调模式相关的命令无效，而与调制解调状态相关的状态位也不会激活。

定时器 / 计数器

定时器 / 计数器 0 到 5 为 5 类时钟输入 ,TIMER_CLOCK1 到 TIMER_CLOCK5。AT91RM9200 中，这些时钟输入与主时钟 (MCK)、慢速时钟 (SLCK) 及主时钟分频后时钟相关，详见 “Clock Control” on page 476。

Table 2 给出定时器/计数器时钟输入与AT91RM9200中时钟的对应关系。每个定时器/计数器0 ~ 5 配置相同。

Table 2. 定时器 / 计数器时钟分配

TC 时钟输入	时钟
TIMER_CLOCK1	MCK/2
TIMER_CLOCK2	MCK/8
TIMER_CLOCK3	MCK/32
TIMER_CLOCK4	MCK/128
TIMER_CLOCK5	SLCK

ARM920T 处理器概述

概述

ARM920T 高缓存处理器是 ARM9™ Thumb® 系列中高性能的 32 位单片系统处理器。它提供完善的高性能 CPU 子系统：

- ARM9TDMI RISC 整数 CPU
- 16-K 字节指令与 16-K 字节数据缓存
- 指令与数据存储器管理单元 (MMUs)
- 写缓冲器
- 高级微处理器总线架构 (AMBA™) 总线接口
- ETM(内置追踪宏单元) 接口

ARM920T中的ARM9TDMI内核可执行32位ARM及16位Thumb指令集。ARM9TDMI处理器是哈佛结构的，有包括取指、译码、执行、存储及写入的五级流水线。

ARM920T 处理器包括两个协处理器：

- CP14 - 控制软件对调试信道的访问。
- CP15 - 系统控制处理器，提供 16 个额外寄存器用来配置与控制缓存、 MMU、系统保护、时钟模式及其他系统选项。

ARM920T 处理器主要特性如下：

- ARM9TDMI®- 内核， ARM® v4T 架构
- 两套指令集
  - ARM 高性能 32 位指令集
  - Thumb 高代码密度 16 位指令集
- 5 级流水线结构
  - 取指 (F)
  - 指令译码 (D)
  - 执行 (E)
  - 数据存储访问 (M)
  - 写寄存器 (W)
- 16-K 字节数据缓存， 16-K 字节指令缓存
  - 虚拟地址 64 路相关缓存
  - 每线 8 字
  - 正向及反向写操作
  - 伪随机或循环置换
  - 低功耗 CAM RAM 设备
- 写缓冲器
  - 16 字的数据缓冲器
  - 4 地址的地址缓冲器
  - 软件控制消耗
- 标准的 ARMv4 存储器管理单元 (MMU)
  - 区域访问许可
  - 允许以 1/4 页面大小对页面进行访问
  - 16 个嵌入域
  - 64 个输入指令 TLB 及 64 个输入数据 TLB



- 8 位、 16 位、 32 位的指令总线与数据总线

方框图

Figure 8. ARM920T 内部功能方框图

ARM920T
Instruction Cache
Instruction MMU
Instruction Physical Address Bus
Instruction Modified Virtual Address Bus
R13
Instruction Virtual Address Bus
Instruction Bus
ICE Interface
ICE
ARM9TDMI
CP15
Bus Interface
Memory Controller
Data Virtual Address Bus
Data Bus
Write Buffer
R13
Data Modified Virtual Address Bus
Data Physical Address Bus
Data Cache
Data MMU
Write Back PA TAG RAM
Write Back Physical Address Bus
Data Index Bus

ARM9TDMI 处理器

指令类型

指令可为 32 位 (ARM 状态中) 或 16 位 (Thumb 状态中)。

数据类型

ARM9TDMI支持字节(8位) ,半字(16位)及字(32位)数据类型。字必须是四字节边界对齐，半字必须是两字节边界对齐。

非对齐数据访问取决与特定区域使用的指令。

ARM9TDMI 工作模式

基于 ARM v4T 架构的 ARM9TDMI 支持 7 种处理器模式：

- 用户模式：标准的 ARM 程序执行状态；
- FIQ：用于支持高速数据传输或通道处理；
- IRQ：用于通用功能中断处理；
- 管理模式：操作系统的保护模式；
- 中止模式：执行虚拟内存与 / 或存储保护；
- 系统模式：操作系统中特许用户模式；
- 未定义模式：支持软件对硬件协处理器的仿真

可通过软件控制、外部中断或异常处理改变模式。绝大部分应用程序在用户模式下执行。当系统响应中断、异常或访问受保护源时进入通常所说的特许模式。



ARM9TDMI 寄存器

ARM9TDMI处理器内核包含一个32位的数据通道及相关控制逻辑。数据通道包括31个通用寄存器、两个全移位器、算术逻辑单元及乘法器。

任何模式下，用户可用 16 个寄存器。其它寄存器用于加速异常处理。

寄存器 15 为程序计数器 (PC)，可用在所有指令中来指示对应于当前指令的数据。

R14 保存子程序调用返回地址。

R13 作为堆栈指针 (软件定义)。

Table 9. ARM9TDMI 模式与寄存器布局

用户与系统模式	管理模式	中止模式	未定义模式	中断模式	快速中断模式
R0	R0	R0	R0	R0	R0
R1	R1	R1	R1	R1	R1
R2	R2	R2	R2	R2	R2
R3	R3	R3	R3	R3	R3
R4	R4	R4	R4	R4	R4
R5	R5	R5	R5	R5	R5
R6	R6	R6	R6	R6	R6
R7	R7	R7	R7	R7	R7
R8	R8	R8	R8	R8	R8_FIQ
R9	R9	R9	R9	R9	R9_FIQ
R10	R10	R10	R10	R10	R10_FIQ
R11	R11	R11	R11	R11	R11_FIQ
R12	R12	R12	R12	R12	R12_FIQ
R13	R13_SVC	R13_ABORT	R13_UNDEF	R13_IRQ	R13_FIQ
R14	R14_SVC	R14_ABORT	R14_UNDEF	R14_IRQ	R14_FIQ
PC	PC	PC	PC	PC	PC

CPSR	CPSR	CPSR	CPSR	CPSR	CPSR
	SPSR_SVC	SPSR_ABORT	SPSR_UNDEF	SPSR_IRQ	SPSR_FIQ

模式列出组寄存器

寄存器 R0 ~ R7 为未分组寄存器，在所有的处理器模式下它们对应相同的 32 位物理寄存器。它们是通用功能寄存器，可用在指令需要通用寄存器的任何场合。

寄存器 R8 ~ R14 为分组寄存器，其定义与当前处理器模式 有关。

更多细节见 ARM 用户手册 Rev. DDI0100E。

模式与异常处理

所有异常处理存入 R14 与 R13 寄存器中。

异常产生后， R14 保存异常处理返回地址。该地址用于异常处理后的返回及对引起异常指令的定址。

R13 用于向每个异常处理提供一个堆栈指针。

快速中断模式中还有寄存器 8 ～ 12，以便开始中断处理时不保存这些寄存器。

系统模式没有固定寄存器。它使用用户模式寄存器。系统模式运行任务时需要一个特许模式并允许它们调用所有类型异常。

状态寄存器

所有其它处理器状态保存在状态寄存器中。当前工作的处理器状态存在当前程序状态寄存器 (CPSR) 中。 CPSR 中保存：

- 四个 ALU 标志 (负数、零、进位及溢出标志) ；
- 两个中断禁用位 (每类中断一个) ；
- 一个 ARM 或 Thumb 运行指示位；
- 五位确定当前处理器模式。

所有的五种异常模式均有一个保存程序状态寄存器 (SPSR)，用于在异常出现前迅速保存 CPSR 中值。

异常类型

ARM9TDMI 支持五种异常类型，每种类型有一个相应的特许处理模式。异常类型为：

- 快速中断 (FIQ)
- 正常中断 (IRQ)
- 存储中止 (用来执行存储器或虚拟存储器的保护)
- 试图执行未定义指令
- 软件中断 (SWI)

异常可由内部或外部源引起。

可同时出现多个异常。

一旦异常出现，对应异常模式的 R14 及 SPSR 用于保存状态。

异常处理结束返回时，SPSR中值移入CPSR，R14中值移入PC。这可通过两种方式实现：

- 将 S 位置位使用数据处理指令，并将 PC 作为目的寄存器。
- 使用多重载入以恢复 CPSR 指令 (LDM)。



ARM 指令集概述

ARM 指令集分为 :

- 分支指令
- 数据处理指令
- 状态寄存器传输指令
- 载入与恢复指令
- 协处理器指令
- 异常产生指令

ARM 指令是有条件执行的。每条指令包含一个 4 位条件码 (位 [31:28])。

详见 ARM920T 技术参考手册 Rev. DDI0151C。

Table 10 给出 ARM 指令助记符列表。

Table 10. ARM 指令助记符列表

助记符	操作
MOV	移动
ADD	加
SUB	减
RSB	反减
CMP	比较
TST	测试
AND	逻辑与
EOR	逻辑异或
MUL	乘
SMULL	有符号长乘
SMLAL	有符号长累乘
MSR	移入状态寄存器
B	分支
BX	分支并交换
LDR	载入字
LDRSH	载入有符号半字
LDRSB	载入有符号字节
LDRH	载入半字
LDRB	载入字节
LDRBT	转化载入寄存器字节
LDRT	转化载入寄存器
LDM	载入乘
SWP	交换字
MCR	移入协处理器
LDC	载入协处理器

助记符	操作
CDP	协处理器数据处理
MVN	移非
ADC	带进位加
SBC	带进位减
RSC	带进位反减
CMN	比较取反
TEQ	测试等价
BIC	位清零
ORR	逻辑 (包括) 或
MLA	累乘
UMULL	无符号长乘
UMLAL	无符号长累乘
MRS	由状态寄存器移出
BL	分支与链接
SWI	软件中断
STR	恢复字
STRH	恢复半字
STRB	恢复字节
STRBT	转化保存寄存器字节
STRT	转化保存寄存器
STM	多路保存
SWPB	交换字节
MRC	由协处理器移出
STC	由协处理器保存

Thumb 指令集概述

Thumb 指令集是 ARM 指令集的一个子集。

Thumb 指令集分为 :

- 分支指令
- 数据处理指令
- 载入与保存指令
- 批量载入与保存指令
- 异常产生指令

Thumb模式下 ,R0 ~ R7八个通用功能寄存器有效。与执行ARM指令时的R0 ~ R7相同。某些 Thumb 指令还访问程序计数器 (ARM 寄存器 15)、链接寄存器 (ARM 寄存器 14) 及堆栈指针 (ARM 寄存器 13)。其他指令对 ARM 寄存器 8 ~ 15 的访问有所限制。

详见 ARM920T 技术参考手册 Rev. DDI0151C。

Table 11 给出 Thumb 指令助记符列表。

Table 11. Thumb 指令助记符列表

助记符	操作	助记符	操作
MOV	移动	MVN	移非
ADD	加	ADC	带进位加
SUB	减	SBC	带进位减
CMP	比较	CMN	比较取反
TST	测试	NEG	去反
AND	逻辑与	BIC	位清零
EOR	逻辑异或	ORR	逻辑 (包括) 或
LSL	逻辑左移	LSR	逻辑右移
ASR	算术右移	ROR	右转
MUL	乘		
B	分支	BL	分支与链接
BX	分支与交换	SWI	软件中断
LDR	载入字	STR	保存字
LDRH	载入半字	STRH	保存半字
LDRB	载入字节	STRB	保存字节
LDRSH	载入有符号半字	LDRSB	载入有符号字节
LDMIA	载入乘	STMIA	多路保存
PUSH	将寄存器推入堆栈	POP	将寄存器推出堆栈





CP15 协处理器

协处理器 15，或系统控制协处理器 CP15，用于处理 ARM9TDMI 的特殊性能，具体如下：

- 片上存储器管理单元 (MMU)
- 指令与 / 或数据缓存
- 写缓冲器

为控制这些特性， CP15 提供了 16 个额外寄存器，见 Table 12。

Table 12. CP15 寄存器

寄存器	名称	访问方式
0	ID 寄存器	只读
1	控制	读 / 写
2	置换表	读 / 写
3	访问控制范围	读 / 写
4	保留位	无
5	默认状态	读 / 写
6	默认地址	读 / 写
7	缓存工作	只写
8	TLB(1) 运行	只写
9	缓存上锁	读 / 写
10	TLB 上锁	读 / 写
11	保留位	无
12	保留位	无
13	FCSE PID(2)	读 / 写
14	保留位	无
15	测试结构	无

Notes: 1. TLB：转换后备缓冲器。
2. FCSE PID：快速前后切换扩展处理标识符。

CP15 寄存器访问

CP15 寄存器只能通过特许模式访问：

- MCR (由 ARM 寄存器移入协处理器) 指令
- MRC (由协处理器移入 ARM 寄存器) 指令

其它指令 (CDP、 LDC、 STC) 产生未定义指令异常。

MCR 指令将 ARM 寄存器值写入 CP15。

MRC 指令将 CP15 值写入 ARM 寄存器。

两指令汇编代码为：

```
MCR/MRC{cond} p15, opcode_1, Rd, CRn, CRm, opcode_2
```

MCR、 MRC 指令位模型如下：

31	30	29	28	27	26	25	24
Cond				1	1	1	0

23	22	21	20	19	18	17	16
opcode_1			L	CRn			

15	14	13	12	11	10	9	8
Rd				1	1	1	1

7	6	5	4	3	2	1	0
opcode_2			1	CRm			

- **CRm[3:0]：特定的协处理器操作**

确定特定的协处理器操作。其值取决于 CP15 寄存器，详见 CP15 指定寄存器性能。

- **opcode_2[7:5]**

确定特定的协处理器操作码，默认情况下设为 0。

- **Rd[15:12]: ARM 寄存器**

定义向协处理器传输值的 ARM 寄存器。若选择 R15，结果无法预测。

- **CRn[19:16]: 协处理器寄存器**

定义目的协处理器寄存器。

- **opcode_1[23:20]: 协处理器代码**

定义协处理器特定码。 CP15 中为 c15。

- **L: 指令位**

0 = MCR 指令

1 = MRC 指令

- **Cond [31:28]: 条件**

存储器管理单元 (MMU)

ARM920T处理器实行增强ARMv4架构，MMU向ARM9TDMI内核指令与地址端口提供转化与许可校验。MMU 通过存储在主存储器中的一套两级页表来控制，提供单地址与转换保护方案。MMU 中的指令与数据 TLB 可单独被锁定与刷新。

Table 13. 详细映射

映射名称	映射大小	访问许可	子页大小
区	1M 字节	区	-
大页	64K 字节	4 个独立子页	16K 字节
小页	4K 字节	4 个独立子页	1K 字节
微页	1K 字节	微页	-

存储域

存储域是存储区与页的集合。ARM920T 支持 16 个存储域。域访问控制寄存器控制对这些域的访问，详见 "CP15 寄存器 3，域访问控制寄存器" on page 52。

MMU 故障

MMU 会产生队列故障、转化故障、存储域故障及存储许可故障。无论 MMU 使能与否均不影响队列故障校验。

MMU 访问控制检查产生这些故障的条件。若故障是由于存储器访问引起的，MMU 中止访问并向 CPU 内核发出出错信号。MMU 将状态与地址故障存储于 FSR 与 FAR 寄存器 (只针对由数据访问产生的故障)。

MMU 不保存由于取指产生的故障信息。

在线取指、存储访问与转换表访问时可中止存储器系统。

缓存，写缓冲器与物理地址

ARM920T 包括一个指令缓存 (ICache)、一个数据缓存 (DCache)、一个写缓冲器与一个物理地址 (PA) TAG RAM 来减轻主存储器带宽与等待时间性能。

ARM920T 实现了 16-K 字节指令与 16-K 字节数据缓存分离。

缓存与写缓冲器由 CP15 寄存器 1(控制)、CP15 寄存器 7(缓存操作) 及 CP15 寄存器 9 (缓存上锁) 控制。

指令缓存 (ICache)

ARM920T含有一个16-K字节指令缓存(ICache)。ICache有512线32字节，排列为64路相关缓存。

指令访问取决于 MMU 许可及转换检测。

若 ICache 使能而 MMU 禁用，所有读取指令均可存入缓存中。物理地址平直映射到修正过的虚拟地址，无保护校验。

当 ICache 禁用，忽略缓存内容，所有取得的指令送到 AMBA 总线上。

复位时，ICache 入口将无效且 ICache 被禁用。为实现最佳性能，复位后应尽快将 ICache 使能。

在 CP15 寄存器 1 中写入 1 将使能 ICache，若写入 0 将禁用 ICache，详见 “CP15 寄存器 1，控制 ” on page 49。

ICache 分为 8 块，每块有 64 条口线，每条口线上又由 8 个字组成。口线在块中的位置称为索引，其序号从 0 到 63。

缓存中的线由索引与块来标识。索引与 MVA (修正虚拟地址) 无关，而块为 MVA 中的位 [7:5]。

数据缓存 (DCache) 与写缓冲器

ARM920T包括一个16-K字节数据缓存(DCache)。DCache有512线32字节，排列为64路相关缓存，通过 ARM9DTMI 内核的 CP15 寄存器 13 使用 MVA 转换。

DCache

DCache分为8块，每块有64条口线，每条口线上又由8个字组成。口线在块中的位置称为索引，其序号从 0 到 63。

写缓冲器最高能保存 16 字数据及 4 个独立地址。

DCache 与写缓冲器操作近似，因为它们被 MMU 转换表分页描述符配置在各块中。

所有数据访问必须通过 MMU 允许及转换校验。若 MMU 不能填满流水线或通过 AMBA ASB 接口访问数据，数据访问将被中止。

写通操作

当数据访问命中缓存时，缓存中包含该数据的线将更新其值。新值将立刻写入主存储器中。

写回操作

当数据访问命中缓存时，缓存线被标记为脏，即其内容将不与主存储器中数据同步更新。

写缓冲器

ARM920T 包括一个 16 入口的写缓冲器以避免执行外部存储器写操作时延迟处理器。当要存储时，其数据、地址及其它细节高速写入写缓冲器中。写缓冲器以主存储器速度完成存储 (一般比 ARM 速度慢)。并行处理中， ARM9TDMI 处理器可全速处理下条指令。

物理地址标识 RAM (PA TAG RAM)

ARM920T使用物理地址标识RAM (PA TAG RAM)来执行对数据缓存的写回。所有在数据缓存中的线物理地址保存在 PA TAG 存储器中，当由缓存中收回该线时，删除其地址。

当线写入数据缓存时，物理地址标识写入 PA TAG RAM。若该线要写回主存储器，由 PA TAG RAM 中读出，使用 AMBA ASB 接口将物理地址写回。

对于 16-K 字节 DCache， PA TAG RAM 由 8 块组成：

- 每块 64 行
- 每行 26 位

ARM920T 用户接口

CP15 寄存器 0 ， ID 代码及缓存类型

访问：只读

CP 寄存器 0 包含详细的硬件信息。读访问内容由 opcode_2 域值确定。对寄存器 0 写入结果无法预计。

ID 代码

将 opcode_2 域置 0 后读寄存器 0 可访问 ID 代码寄存器 。

ID 代码内容如下：

31	30	29	28	27	26	25	24
imp							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
SRev				archi			
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PNumber							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
				Layout Rev			

- **LayoutRev[3:0]: 版本**

包含处理器版本号

- **PNumber[15:4]: 处理器部分序号**

ARM920T 处理器值为 0x920

- **archi[19:16]: 架构**

设备架构详细代码。

ARMv4T 架构值为 0x2

- **SRev[23:20]: 详细版本序号**

详细版本序号用来辨别相同主部分的两个变量，值为 0x1 。

- **imp[31:24]: 设备代码**

0x41 (= A) ；表示 ARM 公司。



缓存类型

当 opcode_2 置为 1 时缓存类型寄存器通过读取寄存器 0 访问。

缓存类型寄存器包含缓存大小与架构信息。

31	30	29	28	27	26	25	24
0	0	0	ctype				S
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
DSize							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ISize							

- **ISize[11:0]: 指令缓存大小**

说明指令缓存的大小、线长组合规则。

- **DSize[23:12]: 数据缓存大小**

说明数据缓存的大小、线长组合规则。

- **S[24]: 缓存**

说明缓存是一体化的还是分为指令缓存与数据缓存。

置位为 1，说明分为指令缓存与数据缓存。

- **ctype[28:25]: 缓存类型**

定义缓存类型。

关于 DSize 与 ISize 位更详细的内容见 ARM920T 技术参考手册 Rev. DDI0151C。

CP15 寄存器 1，控制

访问：读 / 写

CP15 寄存器 1 或控制寄存器包含 ARM920T 控制位

31	30	29	28	27	26	25	24
iA	nF	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	RR	V	I	0	0	R	S

7	6	5	4	3	2	1	0
B	1	1	1	1	C	A	M

- **M[0]: MMU 使能**

0 = MMU 禁用

1 = MMU 使能

- **A[1]: 队列故障使能**

0 = 故障校验禁用

1 = 故障校验使能

- **C[2]: DCache 使能**

0 = DCache 禁用

1 = DCache 使能

- **B[7]: Endianness**

0 = 小 endian 模式

1 = 大 endian 模式

- **S[8]: 系统保护**

修改 MMU 保护系统

详见 ARM920T 技术参考手册 Rev. DDI0151C。

- **R[9]: ROM 保护**

修改 MMU 保护系统

详见 ARM920T 技术参考手册 Rev. DDI0151C。

- **I[12]: ICache 控制**

0 = ICache 禁用

1 = ICache 使能

- **V[13]: 异常寄存器基地址**

0 = 低地址，为 0x00000000

1 = 高地址，为 0xFFFF0000

- **RR[14]: Round Robin 置换**

0 = 随机置换

1 = Round robin 置换



- **时钟模式 [31:30] (iA 与 nF 位)**

iA	nF	时钟模式
0	0	快速总线
0	1	同步
1	0	保留
1	1	异步

CP15 寄存器 2, TTB

访问：读 / 写

CP15 寄存器 2，或转换表基 (TTB) 寄存器，定义转换表第一级

31	30	29	28	27	26	25	24
指针							

23	22	21	20	19	18	17	16
指针							

15	14	13	12	11	10	9	8
指针		-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-

- **指针 [31:14]**

指向第一级转换表基。读返回当前第一级转换表。写设置第一级表指针指向写入值。

当写入时，未定义位为 0，读出时结果无法预见。





CP15 寄存器 3，域访问控制寄存器

访问：读 / 写

CP 15 寄存器 3，或域访问控制寄存器，定义允许域访问。

使用 16 域进行 MMU 访问优先级控制。

寄存器 3 中的每一块对应一个域。

31	30	29	28	27	26	25	24
D15		D14		D13		D12	

23	22	21	20	19	18	17	16
D11		D10		D9		D8	

15	14	13	12	11	10	9	8
D7		D6		D5		D4	

7	6	5	4	3	2	1	0
D3		D2		D1		D0	

- **D15 ~ D0: 域名访问**

2 位值定义了对域的访问，说明见下表：

值		访问	说明
0	0	不访问	访问将产生域故障
0	1	客户	用户域（执行程序，访问数据），域访问许可控制域访问
1	0	保留	保留
1	1	管理器	控制域行为，域访问许可后不校验

CP15 寄存器 4，保留

对该寄存器的访问 (读或写) 结果无法预见。

CP15 寄存器 5，故障状态寄存器

访问：读 / 写

读 CP 15 寄存器 5，或故障状态寄存器 (FSR)，返回最后数据故障源，表示当数据中止出现时尝试访问的域与类型。

此外，将引起数据中止的虚拟地址写入故障地址寄存器 (CP15 寄存器 6)。

写 CP 15 寄存器 5，或故障状态寄存器 (FSR)，设置数据写入时 FSR 值。用于调试器恢复 FSR 中值。

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
域				状态			

- **Status[3:0]: 故障类型**

说明故障类型。当数据中止出现时由 MMU 对状态域编码。状态域译码由域名及与数据中止相关的 MVA(存于 FAR 中) 确定。

- **Domain[7:4]: 域**

说明当故障出现时访问的域 (D15 - D0)。

当写入时，未定义位为 0，读出时结果无法预见。



CP15 寄存器 6，故障地址寄存器

访问：读 / 写

CP 15 寄存器 6，或故障地址寄存器 (FAR)，包含当最后故障出现时尝试访问的 MVA 。FAR 只会因数据故障而更新，不会应预取故障改变。

对 FAR 的写性能，允许调试器保存一个先前状态。

31	30	29	28	27	26	25	24
FAR							

23	22	21	20	19	18	17	16
FAR							

15	14	13	12	11	10	9	8
FAR							

7	6	5	4	3	2	1	0
FAR							

- **FAR[31:0]: 故障地址**

读：返回 FAR 值。 FAR 中保存故障出现时尝试访问的虚拟地址。

写：将 FAR 设置为写入数据值。用于调试器恢复 FSR 中值。

CP15 寄存器 7，缓存工作寄存器

访问：只写

CP15 寄存器 7，或缓存工作寄存器，用以管理指令缓存 (ICache) 与数据缓存 (DCache)。

每个缓存工作功能由 pcode_2 及使用写 CP15 寄存器 7 的 MCR 指令的 CRm 域选定。

Table 14. 缓存功能

功能	数据	CRm	opcode_2
等待中断	SBZ	c0	4
使 ICache 无效	SBZ	c5	0
使 ICache 单入口 (使用 MVA) 无效	MVA 格式	c5	1
使 DCache 无效	SBZ	c6	0
使 DCache 单入口 (使用 MVA) 无效	MVA 格式	c6	1
使 ICache 与 DCache 无效	SBZ	c7	0
清除 DCache 单入口 (使用 MVA)	MVA 格式	c10	1
清除 DCache 单入口 (使用索引)	索引格式	c10	2
耗写缓冲器	SBZ	c10	4
预取 ICache 线 (使用 MVA)	MVA 格式	c13	1
清除并使 DCache 入口无效 (使用 MVA)	MVA 格式	c14	1
清除并使 DCache 入口无效 (使用索引)	索引格式	c14	2

功能详述

- 等待中断

低电压状态停止运行等待中断出现。

- 无效

缓存线标记为无效，使得该线不会被访问命中直到重新分配一个地址。

- 清除

应用于写回数据缓存中。若缓存中有未写出的数据，立即写入主存储器。

- 耗写缓冲器

写缓冲器中的数据完全写入主存储器后停止执行。

- 预取

在指定的虚拟地址中的存储器缓存线载入缓存。

该操作在单缓存线执行，使用 MCR 指令中的数据转移标识缓存线。

数据由下面两种格式中的一种进行解释：

– MVA 格式
– 索引格式

下面是 MVA 格式的 CP15 寄存器 7，或缓存功能寄存器

31	30	29	28	27	26	25	24
mva							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
mva							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
mva							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
mva			-	-	-	-	-

- **mva[31:5]: 修正的虚拟地址**

当写入时，未定义位为 0，读出时结果无法预见。

下面是索引格式的 CP15 寄存器 7，或缓存功能寄存器

31	30	29	28	27	26	25	24
index						-	-
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
-	-	-	-	-	-	-	-
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
-	-	-	-	-	-	-	-
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
set			-	-	-	-	-

- **index[31:26]: 线**

确定缓存线。

- **set[7:5]: 块**

确定缓冲块

当写入时，未定义位为 0，读出时结果无法预见。

写其它 opcode_2 值或 CRm 值结果无法预见。

由 CP15 寄存器 7 读出结果无法预见。

CP15 寄存器 8 ， TLB 工作寄存器

访问 : 只读

CP15 寄存器 8，或转换后备缓冲器 (TLB) 工作寄存器，用于管理指令 TLB 与数据 TLB。

使用 opcode_2 及写 CP15 寄存器 8 的 MCR 指令中的 CRm 域选定 TLB 工作 。

Table 15. TLB 工作

功能	数据	CRm	opcode_2
使 I TLB 无效	SBZ	5	0
使 I TLB 单入口 (使用 MVA) 无效	MVA format	5	1
使 D TLB 无效	SBZ	6	0
使 D TLB 单入口 (使用 MVA) 无效	MVA format	6	1
使指令与数据 TLB 无效	SBZ	7	

下面描述了 TLB 工作在 MVA 格式单入口的 CP15 寄存器 8

31	30	29	28	27	26	25	24
mva							

23	22	21	20	19	18	17	16
mva							

15	14	13	12	11	10	9	8
mva						-	-

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-

- **mva[31:10]: 修正的虚拟地址**

当写入时，未定义位为 0，读出时结果无法预见。

写其它 opcode_2 值或 CRm 值结果无法预见。

由 CP15 寄存器 8 读出结果无法预见。

CP15 寄存器 9，缓存上锁寄存器

访问：读 / 写

CP15寄存器9，或缓存上锁寄存器，复位时值为0x0。缓存上锁寄存器允许软件控制在ICache或DCache上的缓存线上载入填充。防止在填充时 ICache 或 DCache 被驱逐，将其锁定在缓存中。

由 CP15 寄存器 9 读取返回缓存上锁寄存器值，即所有缓存段的基地址指针。

只返回 [31:26]，其它值不可预见。

对 CP15 寄存器 9 写入更新缓存上锁寄存器，所有缓存段基地址与当前地址指针更新。

Table 16. 缓存上锁功能

功能	数据	CRm	opcode_2
读 DCache 上锁基	时基	0	0
写 DCache victim 与上锁基	victim= 时基	0	0
读 ICache 上锁基	时基	0	1
写 ICache victim 与上锁基	victim= 时基	0	1

31	30	29	28	27	26	25	24
index						-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-

- **index[31:26]: Victim 指针**

当前 victim 指针表示将被填充的缓存中的线。

当写入时，未定义位为 0，读出时结果无法预见。

CP15 寄存器 10， TLB 上锁寄存器

访问 : 读 / 写

CP15寄存器10 ,或 TLB上锁寄存器复位时值为0x0。每个TLB均有一个TLSB上锁寄存器；opcode_2值确定访问哪个TLB寄存器：

- opcode_2 = 0x0 ， D TLB 寄存器
- opcode_2 = 0x1 ， I TLB 寄存器

Table 17. TLB 上锁功能

功能	数据	CRm	Opcode_2
读 D TLB 上锁	TLB 上锁	0	0
写 D TLB 上锁	TLB 上锁	0	0
读 I TLB 上锁	TLB 上锁	0	1
写 I TLB 上锁	TLB 上锁	0	1

31	30	29	28	27	26	25	24
Base							

23	22	21	20	19	18	17	16
Victim				-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	P

- **Base[31:26]: 基址**

TLB 替换策略使用的 TLB 入口从基址到 63。提供的当前区域也在这个范围内。

- **Victim[25:20]: Victim 计数器**

列出被覆盖的 TLB 入口。

- **P[0]: 保留**

为 0， TLB 入口无效。

为 1， TLB 入口保护。在所有指令无效时它不能无效，参见 “CP15 寄存器 8， TLB 工作寄存器 ” on page 57。

当写入时，未定义位为 0，读出时结果无法预见。

CP15 寄存器 11， 12，保留

对这些寄存器的访问 (读或写) 结果不可预见。



CP15 寄存器 13 ， FCSE PID 寄存器

访问 : 读 / 写

CP15 寄存器 13，或快速前后切换扩展 (FCSE) 处理标识符 (PID) 寄存器，复位时值为 0x0。

由 CP15 寄存器 13 读取返回 FCSE PID 值。

向 CP15 寄存器 13 写入置位 FCSE PID。

FCSE PID 设置 ARM9TDMI 与缓存存储器 MMU 间映射。

ARM9TDMI 地址范围为 0 ～ 32 M 字节，通过 FCSE PID 转换。

31	30	29	28	27	26	25	24
FCSEPID							-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-

- **FCSEPID[31:25]: FCSE PID**

FCSE PID 修正 ARM920T 存储器系统修改。该修改允许多程序在 ARM 上运行。

4-GB 虚拟地址分为 128 个 32 M 字节的处理模块。每个模块可包含一个编译后使用地址空间范围从 0x00000000 到 0x01FFFFFF 的程序。对每个 i = 0 到 127 的处理模块，i 由地址 i*0x20000000 到地址 i*0x20000000 + 0x01FFFFFF 运行。

详见 ARM920T 技术参考手册 Rev. DDI0151C。

当写入时，未定义位为 0，读出时结果无法预见。

CP15 寄存器 14，保留

对这些寄存器的访问 (读或写) 结果不可预见。

CP15 寄存器 15，测试配置寄存器

CP15 寄存器 15，或测试配置寄存器用于测试。对该寄存器的访问 (读或写) 结果不可预见。

调试与测试特性 (DBG Test)

概述

AT91RM9200有许多起重要作用的互补的调试与测试能力。普通的JTAG/ICE (内部仿真器)端口作为标准调试功能用来下载代码并单步直达程序。 ETM (内置追踪宏单元) 提供更完善的调试性能如地址与数据比较器、半速时钟模式、计数器、序列发生器与 FIFO。调试单元提供一个两引脚 UART 用来将应用程序上载到内部 SRAM 中。它管理用于跟踪工作的调试信道的内部 COMMTX 与 COMMRX 信号中断处理。

一套专用的调试与测试输入 / 输出引脚提供在 PC 测试环境中对这些性能的直接访问。

调试与测试特性如下 :

- 集成了内置仿真器
- 调试单元
  - 两引脚 UART
  - 调试信道
  - 芯片 ID 寄存器
- 内置追踪宏单元 : ETM9 Rev2a
  - 中级实现
  - 半速时钟模式
  - 四对地址比较器
  - 两个数据比较器
  - 八个存储器映射解码器输入
  - 两个计数器
  - 一个序列发生器
  - 一个 18 字节的 FIFO
- 数字引脚通过 IEEE1149.1 JTAG 边界扫描



方框图

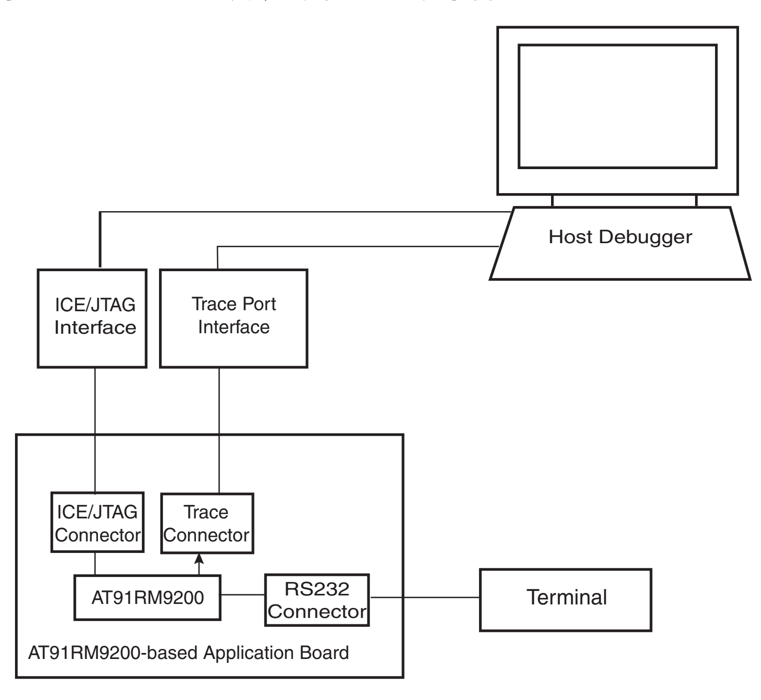
Figure 9. AT91RM9200 调试与测试方框图

应用实例

调试环境

Figure 10 on page 63 给出完整的调试环境示例。ICE/JTAG 接口用于标准的调试功能，如下载代码与单步直达程序。追踪端口接口用于追踪信息。运行在个人电脑上的软件调试器提供利用 ICE/JTAG 接口配置追踪端口接口的用户接口。

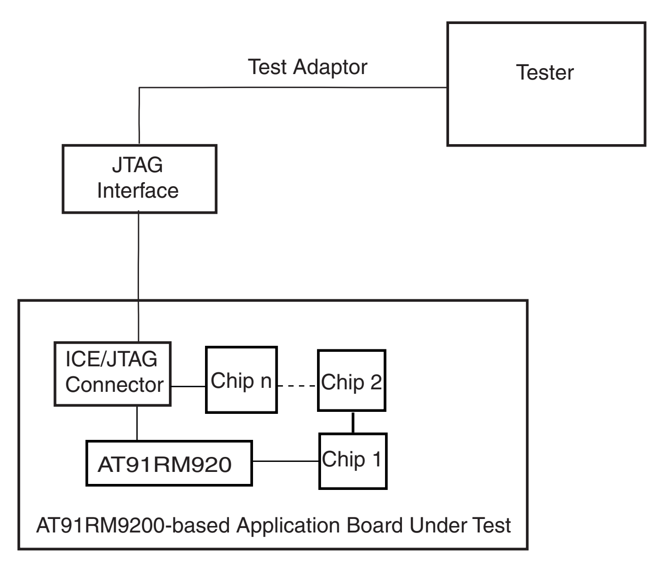
Figure 10. 基于 AT91RM9200 应用调试与追踪环境示例



测试环境

Figure 11 给出测试环境实例。测试向量由测试器发送和解释。本例中，使用许多 适用 JTAG 器件设计 " 测试板 "。这些器件连接起来组成单扫描链。

Figure 11. 基于 AT91RM9200 应用的 IEEE1149.1 测试环境示例







调试与测试引脚说明

Table 18. 调试与测试列表

引脚名称	功能	类型	激活电平
	复位 / 测试		
NRST	微控制器复位	输入	低
TST0	测试模式选择	输入	
TST1	测试模式选择	输入	
	ICE 与 JTAG		
TCK	测试时钟	输入	
TDI	测试数据输入	输入	
TDO	测试数据输出	输出	
TMS	测试模式选择	输入	
NTRST	测试复位信号	输入	低
JTAGSEL	JTAG 选择	输入	
	ETM (只在 BGA 封装中有效)		
TSYNC	追踪同步信号	输出	
TCLK	追踪时钟	输出	
TPS0- TPS2	追踪 ARM 流水线状态	输出	
TPK0 - TPK15	追踪分组端口	输出	
	调试单元		
DRXD	调试接收数据	输入	DRXD
DTXD	调试发送数据	输出	DTXD

功能说明

测试模式引脚

两个专门引脚 (TST1、 TST0) 用来定义器件的测试模式。用户必须确保这些引脚都为低以保证正常的工作条件。与这些引脚相关的其它值保留作厂商测试用。

内置仿真器

ARM9TDMI 通过 ICE/JTAG 端口支持内置仿真器。它通过 ICE 接口与主计算机连接。使用内置 ARM920T 的 ARM9TDMI 内核执行调试。ARM920T 内部状态是通过 ICE/JTAG 端口检测的，可不使用外部数据总线将指令串行插入内核流水线。因此，在调试状态时批量存储指令 (STM) 可以插入指令流水线中。将输出 ARM9TDMI 寄存器中的内容。该数据能在不影响系统其它部分的情况下 串行移出。

ARM920T处理器有六个扫描链支持测试、调试及对内置ICE编程。扫描链通过ICE/JTAG端口控制。

当 JTAGSEL 为低时选择内置 ICE 模式。不能在 ICE 与 JTAG 操作间直接切换。JTAGSEL 改变后必须执行芯片复位 (NRST 与 NTRST)。单独提供内置 ICE (NTRST) 测试复位输入以便于对引导程序的调试。

更多关于内置仿真器的细节见 ARM920T 技术参考手册 ARM Ltd, - DDI 0151C。

调试单元

调试单元提供一个两脚 (DXRD 与 TXRD) UART，可用于多种调试与追踪功能，并为套件可编程解决方案与调试监控器通信提供了一个理想的方案。此外，两外设数据控制器通道连接以最少的处理器时间来完成这些任务的包处理方式。

调试单元同时还管理来自于 ICE 的 COMMTX 与 COMMRX 信号及对工作的调试信道的追踪中断处理。调试单元通过 ICE 接口阻止对系统的访问。

调试单元可用于向内部 SRAM 上载应用程序。当未检测到有效的应用程序时通过引导程序将其激活。用于下载应用程序的协议是 XMODEM。

调试单元芯片 ID 寄存器中有产品版本号及其内部配置。

AT91RM9200 调试单元芯片 ID 为 32 位宽，其值 0x09290781。

关于调试单元更多的内容见 Atmel literature number, 2641 的调试单元。

关于调试单元与引导程序更多的内容见引导程序标准。

内置追踪宏单元

AT91RM9200 有一个与 ARM9TDMI 处理器紧密连接的内置追踪宏单元 (ETM)。内置追踪是标准的中级执行且包括下列源：

- 四对地址比较器
- 两个数据比较器
- 八个存储器映射译码器输入
- 两个计数器
- 一个序列发生器
- 四个外部输入
- 一个内部输出
- 一个 18 字节的 FIFO

AT91RM9200 的内置追踪宏单元工作在半速时钟模式下，并集成了一个时钟分频器。以保证所有追踪端口信号的最高频率不超过 ARM920T 时钟速率的一半。

AT91RM9200 未使用追踪宏单元的输入输出资源。

内置追踪为实时追踪模式，可追踪 ARM9TDMI 指令与数据。

内置追踪调试特性只在 AT91RM9200 BGA 封装中有效。



关于内置追踪宏单元更多的内容见 ETM9 (Rev2a) 技术参考手册 ARM Ltd. -DDI 0157E。

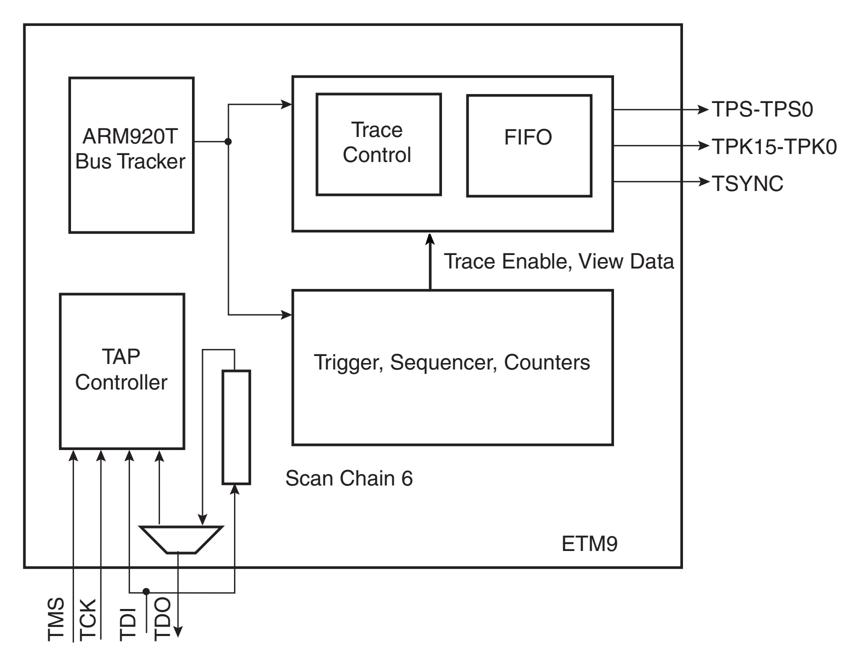
追踪端口

追踪端口由以下引脚构成：

- TSYNC - 同步信号 (表示在追踪分组端口启动分支序列)
- TCLK - 追踪端口时钟， ARM920T 处理器半速时钟
- TPS0 ~ TPS2 - 表示在每个追踪时钟边沿处理器状态
- TPK0 ~ TPK15 - 追踪分组数据值

TPS给出与处理器状态有关的追踪分组信息(地址 ,数据)。某些处理器状态没有与追踪分组端口相关的附加数据 (即指令失败状态代码)。分组长度为 8 位，每个周期输出两个分组。

Figure 12. ETM9 块



执行细节

该节给出内置追踪资源的概述，更多内容见内置追踪宏单元标准 ARM Ltd. -IHI 0014H。

三态序列发生器

序列发生器有三种状态 (一个自检两个对其它发生器检测) 并可在每个时钟周期中改变。状态转换由内部时间控制。若用户需多级触发配置，触发事件基于序列发生器状态。

地址比较器

单模式下，地址比较器将指令地址或数据地址与用户编程地址做比较。

范围模式下，地址比较器成对排列组成虚拟地址范围资源。

关于地址比较器编程细节见：

- 第一个比较器通过范围起始地址编程。
- 第二个比较器通过范围结束地址编程。
- 若地址在下列范围内资源匹配：
  – (地址 > = 范围起始地址) AND (地址 < 范围结束地址)
- 若两比较器未按照相同方式配置，结果无法预见。

数据比较器

每个满地址比较器均与指定的数据比较器相关。数据比较器只有在载入与存储时用来观察数据总线。

数据比较器有一个数值寄存器及一个屏蔽寄存器，因此可以仅将数据总线预编程值与特定位进行比较。

存储译码器输入

八个存储映射译码器输入与用户地址译码器连接。地址译码器将存储器分为片上 SRAM、片上 ROM 及外设。地址译码器同时也优化了 ETM9 追踪触发器。

Table 19. ETM 存储器映射输入安排

说明	区域	访问类型	起始地址	结束地址
SRAM	内部	数据	0x00000000	0x000FFFFF
SRAM	内部	取指	0x00000000	0x000FFFFF
ROM	内部	数据	0x00100000	0x001FFFFF
ROM	内部	取指	0x00100000	0x001FFFFF
NCS0-NCS7	外部	数据	0x10000000	0x8FFFFFFF
NCS0-NCS7	外部	取指	0x10000000	0x8FFFFFFF
用户外设	内部	数据	0xF0000000	0xFFFFEFFF
系统外设	内部	数据	0xFFFFF000	0xFFFFFFFF

FIFO

18 字节 FIFO i 用于保存数据追踪信息。FIFO 将流水线状态由追踪分组中分离出来。因此 ,FIFO 可以用来缓冲追踪包。

当 FIFO 满或当 FIFO 拥有的字节少于用户编程数据时 ,内置追踪宏单元将检测到 FIFO 的溢出。

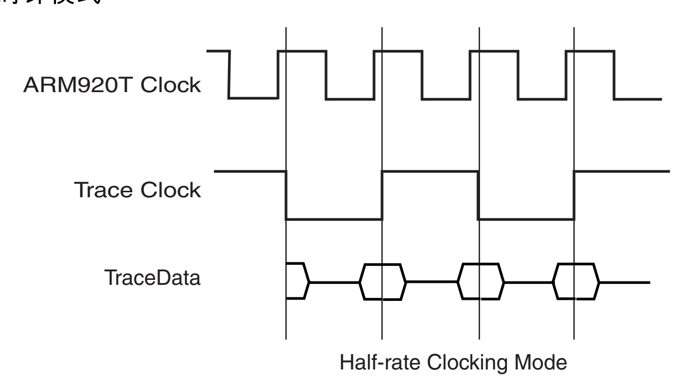
详见 ETM9 (Rev2a) 技术参考手册 ARM Ltd. DDI 0157E。

半速时钟模式

ETM9 运行半速时钟模式时允许在追踪时钟上升沿与下降沿对数据进行追踪。

执行半速模式以维持信号时钟在高速系统中的完整性 (达到 100 Mhz)。

Figure 13. 半速时钟模式



由于需要支持半速时钟功能，选择追踪捕获系统时必须小心。

应用范围限制

由于需要一些定时参数，设置 TCLK 信号时必须小心。

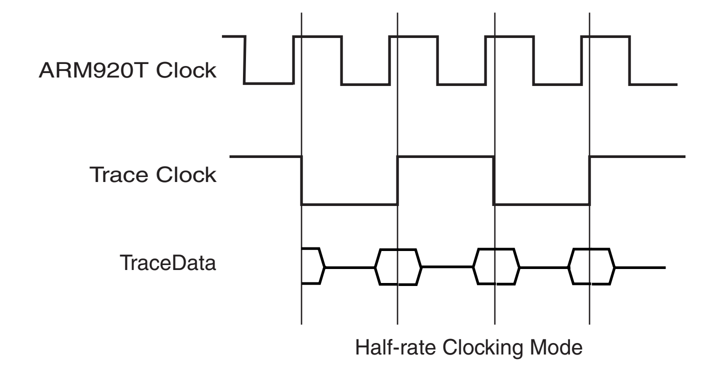
参加 AT91RM9200 “JTAG/ICE Timings” on page 621 及 “ETM Timings” on page 624。

指定的目标系统连接器为 AMP Mictor 连接器。

连接器必须在如 Figure 14 所示的应用范围内使用。图中显示了安装在接近板边缘的追踪连接器的 PCB 正面图。这使得对追踪端口分析仪与连接目标的物理干扰降为最小。



Figure 14. AMP Mictor 连接器校准



IEEE 1149.1 JTAG 边界扫描

IEEE 1149.1 JTAG 边界扫描引脚电平访问独立于器件包技术。

当 JTAGSEL 为高时使能 IEEE 1149.1 JTAG 边界扫描。执行 SAMPLE、EXTEST 与 BYPASS 功能。在 ICE 调试模式下， ARM 处理器以一个非 JTAG 芯片 ID 标识处理器来响应 ICE 系统。在 IEEE 1149.1 JTAG 中不适用。

不能在 JTAG 与 ICE 操作间直接切换。JTAGSEL 改变后必须执行芯片复位 (NRST 与 NTRST)。

两个边界扫描描述语言 (BSDL) 文件用来建立测试。每个 BSDL 文件对应一个特定包。

JTAG 边界扫描寄存器

边界扫描寄存器 (BSR) 包含 449 位，对应于激活引脚及相关控制信号。

每个 AT91RM9200 输入引脚在边界扫描寄存器中有对应位用以观察。

每个 AT91RM9200 输出在 BSR 中对应 2 位寄存器。包含数据的输出位强制在焊盘上。 CTRL 位可将焊盘阻抗拉高。

每个 AT91RM9200 输入 / 输出对应 BSR 中的一个 3 位寄存器。包含数据的输出位强制在焊盘上。输入位便于观察提供到焊盘的数据。 CTRL 位可将焊盘阻抗拉高。

Table 20. JTAG 边界扫描寄存器

比特序号	引脚名称	引脚类型	相关 BSR 单元
449	A19	输出	OUTPUT
448	A[19:16]/BA0/BA1	输出	CTRL
447	A20	输出	OUTPUT
446	A[22:20]/NWE/NWR0	输出	CTRL
445	A21	输出	OUTPUT
444	A22	输出	OUTPUT
443	PC7/A23	I/O	INPUT
442			OUTPUT
441			CTRL
440	PC8/A24	I/O	INPUT
439			OUTPUT
438			CTRL
437	PC9/A25/CFRNW	I/O	INPUT
436			OUTPUT
435			CTRL
434	NCS0/BFCS	输出	OUTPUT
433	NCS[1:0]/NOE/NRD/NUB/ NWR1/NBS1/BFCS/SDCS	输出	CTRL
432	NCS1/SDCS	输出	OUTPUT
431	NCS2	输出	OUTPUT
430	NCS[2:3]/NBS3	输出	CTRL
429	NCS3	输出	OUTPUT
428	NOE/NRD	输出	OUTPUT



Table 20. JTAG 边界扫描寄存器

比特序号	引脚名称	引脚类型	相关 BSR 单元
427	NWE/NWR0	输出	INPUT
426			OUTPUT
425	NUB/NWR1/NBS1	输出	INPUT
424			OUTPUT
423	NBS3	输出	OUTPUT
422	SDCKE	输出	OUTPUT
421	SDCKE/RAS/CAS/WE/SDA10	输出	CTRL
420	RAS	输出	OUTPUT
419	CAS	输出	OUTPUT
418	WE	输出	OUTPUT
417	D0	I/O	INPUT
416			OUTPUT
415	D[3:0]	I/O	CTRL
414	D1	I/O	INPUT
413			OUTPUT
412	D2	I/O	INPUT
411			OUTPUT
410	D3	I/O	INPUT
409			OUTPUT
408	D4	I/O	INPUT
407			OUTPUT
406	D[7:4]	I/O	CTRL
405	D5	I/O	INPUT
404			OUTPUT
403	D6	I/O	INPUT
402			OUTPUT
401	D7	I/O	INPUT
400			OUTPUT
399	D8	I/O	INPUT
398			OUTPUT
397	D[11:8]	I/O	CTRL
396	D9	I/O	INPUT
395			OUTPUT
394	D10	I/O	INPUT
393			OUTPUT

Table 20. JTAG 边界扫描寄存器

比特序号	引脚名称	引脚类型	相关 BSR 单元
392	D11	I/O	INPUT
391			OUTPUT
390	D12	I/O	INPUT
389			OUTPUT
388	D[15:12]	I/O	CTRL
387	D13	I/O	INPUT
386			OUTPUT
385	D14	I/O	INPUT
384			OUTPUT
383	D15	I/O	INPUT
382			OUTPUT
381	PC16/D16	I/O	INPUT
380			OUTPUT
379			CTRL
378	PC17D17	I/O	INPUT
377			OUTPUT
376			CTRL
375	PC18/D18	I/O	INPUT
374			OUTPUT
373			CTRL
372	PC19/D19	I/O	INPUT
371			OUTPUT
370			CTRL
369	PC20/D20	I/O	INPUT
368			OUTPUT
367			CTRL
366	PC21/D21	I/O	INPUT
365			OUTPUT
364			CTRL
363	PC22/D22	I/O	INPUT
362			OUTPUT
361			CTRL
360	PC23/D23	I/O	INPUT
359			OUTPUT
358			CTRL



Table 20. JTAG 边界扫描寄存器

比特序号	引脚名称	引脚类型	相关 BSR 单元
357	PC24/D24	I/O	INPUT
356			OUTPUT
355			CTRL
354	PC25/D25	I/O	INPUT
353			OUTPUT
352			CTRL
351	PC26/D26	I/O	INPUT
350			OUTPUT
349			CTRL
348	PC27/D27	I/O	INPUT
347			OUTPUT
346			CTRL
345	PC28/D28	I/O	INPUT
344			OUTPUT
343			CTRL
342	PC29/D29	I/O	INPUT
341			OUTPUT
340			CTRL
339	PC30/D30	I/O	INPUT
338			OUTPUT
337			CTRL
336	PC31/D31	I/O	INPUT
335			OUTPUT
334			CTRL
333	PC10/NCS4/CFCS	I/O	INPUT
332			OUTPUT
331			CTRL
330	PC11/NCS5/CFCE1	I/O	INPUT
329			OUTPUT
328			CTRL
327	PC12/NCS6/CFCE2	I/O	INPUT
326			OUTPUT
325			CTRL
324	PC13/NCS7	I/O	INPUT
323			OUTPUT
322			CTRL

Table 20. JTAG 边界扫描寄存器

比特序号	引脚名称	引脚类型	相关 BSR 单元
321	PC14	I/O	INPUT
320			OUTPUT
319			CTRL
318	PC15	I/O	INPUT
317			OUTPUT
316			CTRL
315	PC0/BCFK	I/O	INPUT
314			OUTPUT
313			CTRL
312	PC1/BFRDY/SMOE	I/O	INPUT
311			OUTPUT
310			CTRL
309	PC2/BFAVD	I/O	INPUT
308			OUTPUT
307			CTRL
306	PC3/BFBAA/SMWE	I/O	INPUT
305			OUTPUT
304			CTRL
303	PC4/BFOE	I/O	INPUT
302			OUTPUT
301			CTRL
300	PC5/BFWE	I/O	INPUT
299			OUTPUT
298			CTRL
297	PC6/NWAIT	I/O	INPUT
296			OUTPUT
295			CTRL
294	PA0/MISO/PCK3	I/O	INPUT
293			OUTPUT
292			CTRL
291	PA1/MOSI/PCK0	I/O	INPUT
290			OUTPUT
289			CTRL
288	PA2/SPCK/IRQ4	I/O	INPUT
287			OUTPUT
286			CTRL



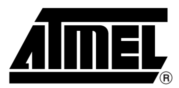


Table 20. JTAG 边界扫描寄存器

比特序号	引脚名称	引脚类型	相关 BSR 单元
285	PA3/NPCS0/IRQ5	I/O	INPUT
284			OUTPUT
283			CTRL
282	PA4/NPCS1/PCK1	I/O	INPUT
281			OUTPUT
280			CTRL
279	PA5/NPCS2/TXD3	I/O	INPUT
278			OUTPUT
277			CTRL
276	PD0/ETX0	I/O	INPUT
275			OUTPUT
274			CTRL
273	PD1/ETX1	I/O	INPUT
272			OUTPUT
271			CTRL
270	PD2/ETX2	I/O	INPUT
269			OUTPUT
268			CTRL
267	PD3/ETX3	I/O	INPUT
266			OUTPUT
265			CTRL
264	PD4/ETXEN	I/O	INPUT
263			OUTPUT
262			CTRL
261	PD5/ETXER	I/O	INPUT
260			OUTPUT
259			CTRL
258	PD6/DTXD	I/O	INPUT
257			OUTPUT
256			CTRL
255	PA6/NPCS3/RXD3	I/O	INPUT
254			OUTPUT
253			CTRL
252	PA7/ETXCK/EREFCK/PCK2	I/O	INPUT
251			OUTPUT
250			CTRL

Table 20. JTAG 边界扫描寄存器

比特序号	引脚名称	引脚类型	相关 BSR 单元
249	PA8/ETXEN/MCCDB	I/O	INPUT
248			OUTPUT
247			CTRL
246	PA9/ETX0/MCDB0	I/O	INPUT
245			OUTPUT
244			CTRL
243	PA10/ETX1/MCDB1	I/O	INPUT
242			OUTPUT
241			CTRL
240	PA11/ECRS/ECRSDV/MCDB2	I/O	INPUT
239			OUTPUT
238			CTRL
237	PA12/ERX0/MCDB3	I/O	INPUT
236			OUTPUT
235			CTRL
234	PA13/ERX1/TCLK0	I/O	INPUT
233			OUTPUT
232			CTRL
231	PA14/ERXER/TCLK1	I/O	INPUT
230			OUTPUT
229			CTRL
228	PA15/EMDC/TCLK2	I/O	INPUT
227			OUTPUT
226			CTRL
225	PA16/EMDIO/IRQ6	I/O	INPUT
224			OUTPUT
223			CTRL
222	PA17/TXD0/TIOA0	I/O	INPUT
221			OUTPUT
220			CTRL
219	PA18/RXD0/TIOB0	I/O	INPUT
218			OUTPUT
217			CTRL
216	PA19/SCK0/TIOA1	I/O	INPUT
215			OUTPUT
214			CTRL



Table 20. JTAG 边界扫描寄存器

比特序号	引脚名称	引脚类型	相关 BSR 单元
213	PA20/CTS0/TIOB1	I/O	INPUT
212			OUTPUT
211			CTRL
210	PA21/RTS0/TIOA2	I/O	INPUT
209			OUTPUT
208			CTRL
207	PA22/RXD2/TIOB2	I/O	INPUT
206			OUTPUT
205			CTRL
204	PA23/TXD2/IRQ3	I/O	INPUT
203			OUTPUT
202			CTRL
201	PA24/SCK2/PCK1	I/O	INPUT
200			OUTPUT
199			CTRL
198	PA25/TWD/IRQ2	I/O	INPUT
197			OUTPUT
196			CTRL
195	PA26/TWCK/IRQ1	I/O	INPUT
194			OUTPUT
193			CTRL
192	PA27/MCCK/TCLK3	I/O	INPUT
191			OUTPUT
190			CTRL
189	PA28/MCCDA/TCLK4	I/O	INPUT
188			OUTPUT
187			CTRL
186	PA29/MCDA0/TCLK5	I/O	INPUT
185			OUTPUT
184			CTRL
183	PA30/DRXD/CTS2	I/O	INPUT
182			OUTPUT
181			CTRL
180	PA31/DTXD/RTS2	I/O	INPUT
179			OUTPUT
178			CTRL

Table 20. JTAG 边界扫描寄存器

比特序号	引脚名称	引脚类型	相关 BSR 单元
177	PB0/TF0/RTS3	I/O	INPUT
176			OUTPUT
175			CTRL
174	PB1/TK0/CTS3	I/O	INPUT
173			OUTPUT
172			CTRL
171	PB2/TD0/SCK3	I/O	INPUT
170			OUTPUT
169			CTRL
168	PB3/RD0/MCDA1	I/O	INPUT
167			OUTPUT
166			CTRL
165	PB4/RK0/MCDA2	I/O	INPUT
164			OUTPUT
163			CTRL
162	PB5/RF0/MCDA3	I/O	INPUT
161			OUTPUT
160			CTRL
159	PB6/TF1/TIOA3	I/O	INPUT
158			OUTPUT
157			CTRL
156	PB7/TK1/TIOB3	I/O	INPUT
155			OUTPUT
154			CTRL
153	PB8/TD1/TIOA4	I/O	INPUT
152			OUTPUT
151			CTRL
150	PB9/RD1/TIOB4	I/O	INPUT
149			OUTPUT
148			CTRL
147	PB10/RK1/TIOA5	I/O	INPUT
146			OUTPUT
145			CTRL
144	PB11/RF1/TIOB5	I/O	INPUT
143			OUTPUT
142			CTRL



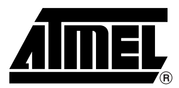


Table 20. JTAG 边界扫描寄存器

比特序号	引脚名称	引脚类型	相关 BSR 单元
141	PB12/TF2/ETX2	I/O	INPUT
140			OUTPUT
139			CTRL
138	PB13/TK2/ETX3	I/O	INPUT
137			OUTPUT
136			CTRL
135	PB14/TD2/ETXER	I/O	INPUT
134			OUTPUT
133			CTRL
132	PB15/RD2/ERX2	I/O	INPUT
131			OUTPUT
130			CTRL
129	PB16/RK2/ERX3	I/O	INPUT
128			OUTPUT
127			CTRL
126	PD7/PCK0/TSYNC	I/O	INPUT
125			OUTPUT
124			CTRL
123	PD8/PCK1/TCLK	I/O	INPUT
122			OUTPUT
121			CTRL
120	PD9/PCK2/TPS0	I/O	INPUT
119			OUTPUT
118			CTRL
117	PD10/PCK3/TPS1	I/O	INPUT
116			OUTPUT
115			CTRL
114	PD11/TPS2	I/O	INPUT
113			OUTPUT
112			CTRL
111	PD12/TPK0	I/O	INPUT
110			OUTPUT
109			CTRL
108	PB17/RF2/ERXDV	I/O	INPUT
107			OUTPUT
106			CTRL

Table 20. JTAG 边界扫描寄存器

比特序号	引脚名称	引脚类型	相关 BSR 单元
105	PB18/RI1/ECOL	I/O	INPUT
104			OUTPUT
103			CTRL
102	PB19/DTR1/ERXCK	I/O	INPUT
101			OUTPUT
100			CTRL
99	PB20/TXD1	I/O	INPUT
98			OUTPUT
97			CTRL
96	PB21/RXD1	I/O	INPUT
95			OUTPUT
94			CTRL
93	PB22/SCK1	I/O	INPUT
92			OUTPUT
91			CTRL
90	PD13/TPK1	I/O	INPUT
89			OUTPUT
88			CTRL
87	PD14/TPK2	I/O	INPUT
86			OUTPUT
85			CTRL
84	PD15/TD0/TPK3	I/O	INPUT
83			OUTPUT
82			CTRL
81	PB23/DCD1	I/O	INPUT
80			OUTPUT
79			CTRL
78	PB24/CTS1	I/O	INPUT
77			OUTPUT
76			CTRL
75	PB25/DSR1/EF100	I/O	INPUT
74			OUTPUT
73			CTRL
72	PB26/RTS1	I/O	INPUT
71			OUTPUT
70			CTRL



Table 20. JTAG 边界扫描寄存器

比特序号	引脚名称	引脚类型	相关 BSR 单元
69	PB27/PCK0	I/O	INPUT
68			OUTPUT
67			CTRL
66	PD16/TD1/TPK4	I/O	INPUT
65			OUTPUT
64			CTRL
63	PD17/TD2/TPK5	I/O	INPUT
62			OUTPUT
61			CTRL
60	PD18/NPCS1/TPK6	I/O	INPUT
59			OUTPUT
58			CTRL
57	PD19/NPCS2/TPK7	I/O	INPUT
56			OUTPUT
55			CTRL
54	PD20/NPCS3/TPK8	I/O	INPUT
53			OUTPUT
52			CTRL
51	PD21/RTS0/TPK9	I/O	INPUT
50			OUTPUT
49			CTRL
48	PD22/RTS1/TPK10	I/O	INPUT
47			OUTPUT
46			CTRL
45	PD23/RTS2/TPK11	I/O	INPUT
44			OUTPUT
43			CTRL
42	PD24/RTS3/TPK12	I/O	INPUT
41			OUTPUT
40			CTRL
39	PD25/DTR1/TPK13	I/O	INPUT
38			OUTPUT
37			CTRL
36	PD26/TPK14	I/O	INPUT
35			OUTPUT
34			CTRL

Table 20. JTAG 边界扫描寄存器

比特序号	引脚名称	引脚类型	相关 BSR 单元
33	PD27/TPK15	I/O	INPUT
32			OUTPUT
31			CTRL
30	PB28/FIQ	I/O	INPUT
29			OUTPUT
28			CTRL
27	PB29/IRQ0	I/O	INPUT
26			OUTPUT
25			CTRL
24	A0/NLB/NBS0	输出	OUPUT
23	A[3:0]/NLB/NWR2/NBS0 /NBS2	输出	CTRL
22	A1/NWR2/NBS2	输出	OUTPUT
21	A2	输出	OUTPUT
20	A3	输出	OUTPUT
19	A4	输出	OUTPUT
18	A[7:4]	输出	CTRL
17	A5	输出	OUTPUT
16	A6	输出	OUTPUT
15	A7	输出	OUTPUT
14	A8	输出	OUTPUT
13	A[11:8]	输出	CTRL
12	A9	输出	OUTPUT
11	A10	输出	OUTPUT
10	SDA10	输出	OUTPUT
9	A11	输出	OUTPUT
8	A12	输出	OUTPUT
7	A[15:12]	输出	CTRL
6	A13	输出	OUTPUT
5	A14	输出	OUTPUT
4	A15	输出	OUTPUT
3	A16/BA0	输出	OUTPUT
2	A17/BA1	输出	OUTPUT
1	A18	输出	OUTPUT



AT91RM9200 ID 代码寄存器

访问 : 只读

31	30	29	28	27	26	25	24
VERSION				PART NUMBER			
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PART NUMBER							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PART NUMBER				MANUFACTURER IDENTITY			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
MANUFACTURER IDENTITY							1

VERSION[31:28]: 产品版本号

置 0x0 = JTAGSEL 为低

置 0x1 = JTAGSEL 为高

PART NUMBER[27:14]: 产品部分号

置为 0x5b02

MANUFACTURER IDENTITY[11:1]

置为 0x01f

Bit [0]: 要求 IEEE Std. 1149.1

置为 1。

AT91RM9200 ID 代码值为 0x15b0203f (JTAGSEL 为高)。

AT91RM9200 ID 代码值为 0x05b0203f (JTAGSEL 为低)。

引导程序

概述

引导程序将应用程序下载到 AT91 集成的 ROM 中。它集成一个 Bootloader 及一个引导 Uploader 以保证正确的信息下载。

先激活 Bootloader。它查找连接在 SPI 上的 DataFlash、连接在两线接口 (TWI) 上的 EEPROM 或连接在外部总线接口 (EBI) 上的 8 位存储器器件上的 8 位有效 ARM 异常向量。除第六条指令外，其余向量必须时 B-branch 或 LDR 载入寄存器。该向量用于保存信息，如载入代码的大小及 DataFlash 器件类型。

若发现有效序列，代码将载入内部 SRAM。这在重映射与跳转到 SRAM 开始地址后进行。

若未发现有效 ARM 向量，引导 Uploader 启动。它将调试单元串行端口 (DBGU) 与 USB 器件端口初始化，然后等待通过 USB 的器件固件升级 (DFU) 协议及 DBGU 的 XMODEM 协议将转换与载入代码载入内部 SRAM。在下载完成后，跳转到 SRAM 开始地址的应用程序入口处。

引导程序的主要特性如下：

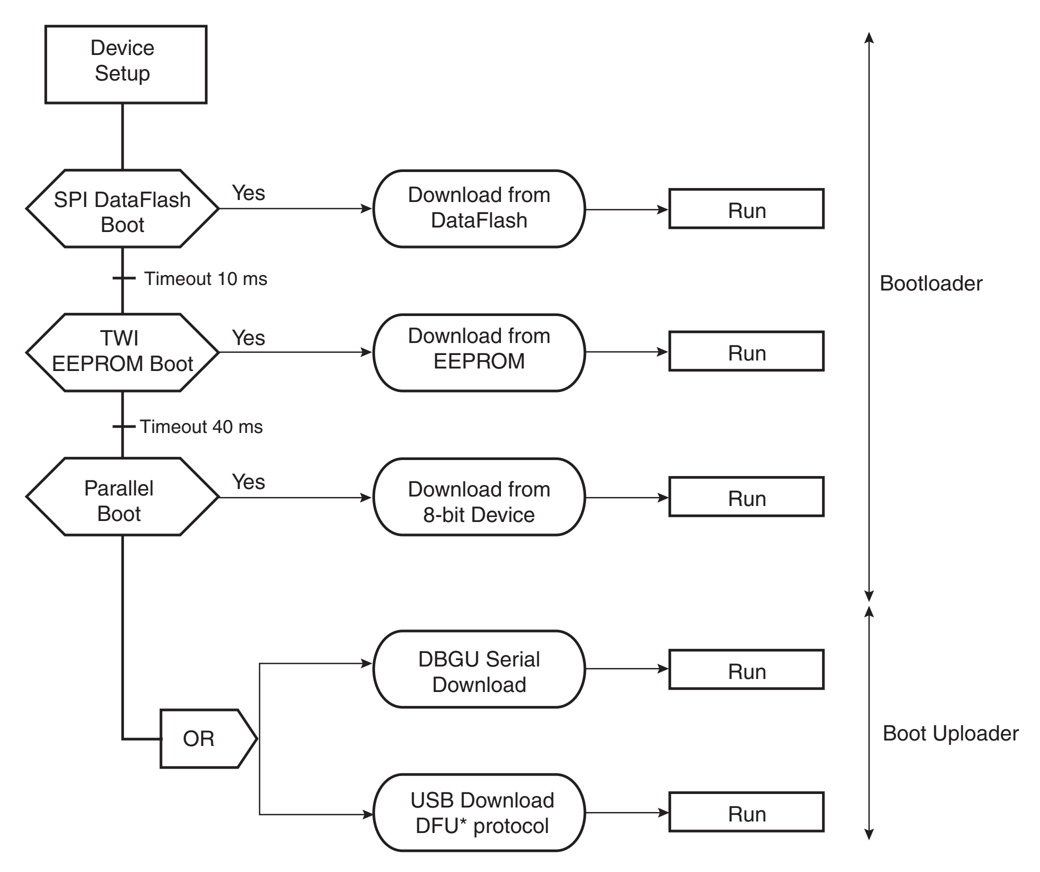
- 默认引导程序存入 ROM 中
- 将应用程序由外部存储媒介载入内部 SRAM 中并在其中运行
- 下载代码大小由内置 SRAM 大小决定
- 有效应用程序自动检测
- Bootloader 支持大量非易失性存储器
  - SPI DataFlash® 连接在 SPI NPCS0 上
  - 两线 EEPROM
  - NCS0 上 8 位并行存储器 (若器件集成 EBI)
- 在外部 NVM 中未检测到可用应用程序时使用引导 Uploader，并支持多种通信媒体
- DBGU 上支持串行通信 (XModem 协议)
- USB 器件端口 (DFU 协议)



流程图

Figure 15 中给出引导程序执行算法。

Figure 15. 引导程序算法流程图



Bootloader

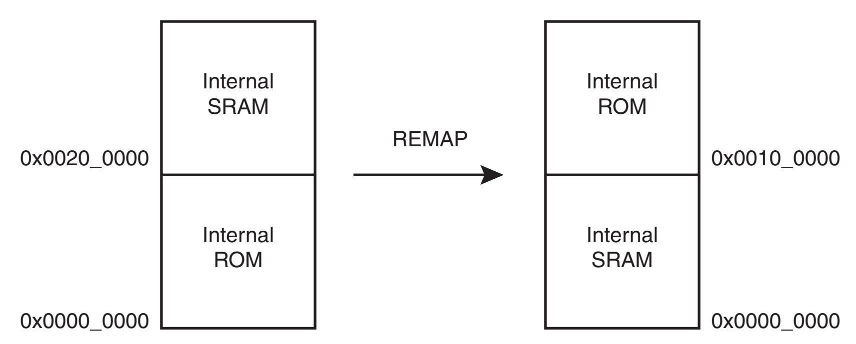
当器件集成复位 EBI 时，若选择片内引导模式，引导程序由地址 0x0000_0000(ARM 复位向量) 开始。第一步是在片外非易失性存储器中搜索有效程序。若发现有效程序，该程序载入内部 SRAM 并在重新映射后跳转到地址 0x0000_0000 开始执行。该程序应是应用代码或二级 Bootloader。

内置引导程序内置了很多可重复使用的功能以优化下载应用程序代码大小。引导程序链接到地址 0x0010_0000 处但内部 ROM 同样及复位后均映射到 0x0000_0000 处。所有的函数调用均与 PC 有关而且不使用绝对地址。 ARM 向量位于 0x0000_0000 与 0x0010_0000 处。

为访问 ROM 中函数，在 ROM 中固定地址定义了一个包含芯片说明与功能函数入口点的结构。

若未找到合适的应用程序，调试串行端口或 USB 器件端口必须连接以允许上载。 Atmel(DFU uploader) 提供的应用程序通过 USB 将应用程序载入内部 SRAM 中。为通过调试串行端口载入应用程序，需要在终端应用程序 (超级终端) 中运行 Xmodem 协议。

Figure 16. 载入完成后重新映射



复位后，内部 ROM 代码映射到地址 0x0000_0000 与 0x0010_0000 处：

```
100000   ea00000b   B     0x2c              00   ea00000b   B     0x2c
100004   e59ff014   LDR   PC,[PC,20]        04   e59ff014   LDR   PC,[PC,20]
100008   e59ff014   LDR   PC,[PC,20]        08   e59ff014   LDR   PC,[PC,20]
10000c   e59ff014   LDR   PC,[PC,20]        0c   e59ff014   LDR   PC,[PC,20]
100010   e59ff014   LDR   PC,[PC,20]        10   e59ff014   LDR   PC,[PC,20]
100014   00001234   LDR   PC,[PC,20]        14   00001234   LDR   PC,[PC,20]
100018   e51fff20   LDR   PC,[PC,-0xf20]    18   e51fff20   LDR   PC,[PC,-0xf20]
10001c   e51fff20   LDR   PC,[PC,-0xf20]    1c   e51fff20   LDR   PC,[PC,-0xf20]
```



有效映射检测

Bootloader通过分析对应于ARM异常向量的头 32字节来软件查找有效应用程序。这些字节必须执行跳转或载入与 PC 寻址相关的 PC ARM 指令。偏移地址为 0x18 的第六个向量包含下载映射大小与 DataFlash 参数。

用户必须用本身向量取代该向量。

Figure 17. LDR Opcode

31			28	27			24	23			20	19 16	15 12	11 0
1	1	1	0	1	1	I	P	U	1	W	0	Rn	Rd	

Figure 18. B Opcode

31			28	27			24	23 0
1	1	1	0	1	0	1	0	Offset (24 bits)

无条件指令：位 31 ~ 28 为 0xE

将与 PC 相关地址载入 PC 指令：

- Rn = Rd = PC = 0xF
- I==1
- P==1
- U 偏移地址 (U==1) 或减 (U==0)
- W==1

示例

有效向量示例：

```
00      ea00000b    B       0x2c
004     e59ff014    LDR     PC, [PC,20]
08      e59ff014    LDR     PC, [PC,20]
0c      e59ff014    LDR     PC, [PC,20]
10      e59ff014    LDR     PC, [PC,20]
14      00001234    LDR     PC, [PC,20]     <- 代码大小 = 4660 字节
18      e51fff20    LDR     PC, [PC,-0xf20]
1c      e51fff20    LDR     PC, [PC,-0xf20]
```

下载模式下 (DataFlash， EEPROM 或集成了 EBI 的 8 位存储器器件)，载入到 SRAM 中的代码大小在第六个 ARM 向量的位置。故用户必须用自身的向量取代当前向量。

ARM 向量 6 结构

ARM 异常向量 6 用来保存引导 ROM downloader 所需的信息，该信息说明如下：

Figure 19. ARM 向量 6 结构

31 17	16 13	12 8	7 0
DataFlash page size	Number of pages	Reserved	Nb of 512 bytes blocks to download

开始的 8 位包含要下载块的数目。块大小为 512 字节，最多允许下载 128K 字节。

位 13 到 16 决定 DataFlash 页数。

– DataFlash 页数 = $2^{(\text{Nb of pages})}$

最后 15 位为 DataFlash 页大小。

Table 21. DataFlash 器件

器件	密度	页大小 (字节)	页数
AT45DB011B	1 Mbit	264	512
AT45DB021B	2 Mbits	264	1024
AT45DB041B	4 Mbits	264	2048
AT45DB081B	8 Mbits	264	4096
AT45DB161B	16 Mbits	528	4096
AT45DB321B	32 Mbits	528	8192
AT45DB642	64 Mbits	1056	8192
AT45DB1282	128 Mbits	1056	16384
AT45DB2562	256 Mbits	2112	16384

示例

以下向量包含描述 AT45DB642 DataFlash 信息，有 11776 字节等待下载。

向量 6 为 0x0841A017 (00001000010000011010000000010111b):

下载大小：0x17 * 512 字节 = 11776 字节

页数 (1101b)：13 ==> DataFlash 页数 = 2^{13} = 8192

DataFlash 页大小 (000010000100000b) = 1056

对 EEPROM 或 8 位外部存储器下载 (若器件集成 EBI)，仅需下载要译码的大小。

Bootloader 序列

引导程序执行初始化后，开始下载程序处理。若失败，通过 USB 或调试串行端口完成上载。

器件初始化

按照以下步骤初始化：

1. PLL 设置
   – PLLB 初始化来产生一个使用 USB 器件所需的 48 MHz 时钟。位于电源管理控制器 (PMC) 的寄存器确定主振荡器频率与 PLLB 正确参数。
   Table 22 定义引导程序支持的晶振值。

Table 22. 软件自动检测支持晶振 (MHz)

3.0	3.2768	3.6864	3.84	4.0
4.433619	4.9152	5.0	5.24288	6.0
6.144	6.4	6.5536	7.159090	7.3728
7.864320	8.0	9.8304	10.0	11.05920
12.0	12.288	13.56	14.31818	14.7456
16.0	17.734470	18.432	20.0	24.0
25.0	28.224	32.0	33.0	

2. 每个 ARM 模式堆栈设置
3. 主振荡器频率检测
4. 中断控制器设置
5. C 变量初始化
6. 跳转主函数

下载程序

下载程序在几个器件中查找有效的引导程序。首先检测的是连接于 SPI 的 NPCS0 上的串行 DataFlash ,然后是连接于TWI的串行EEPROM及外部总线接口的NCS0上的8位并行存储器(若产品中有 EBI)。

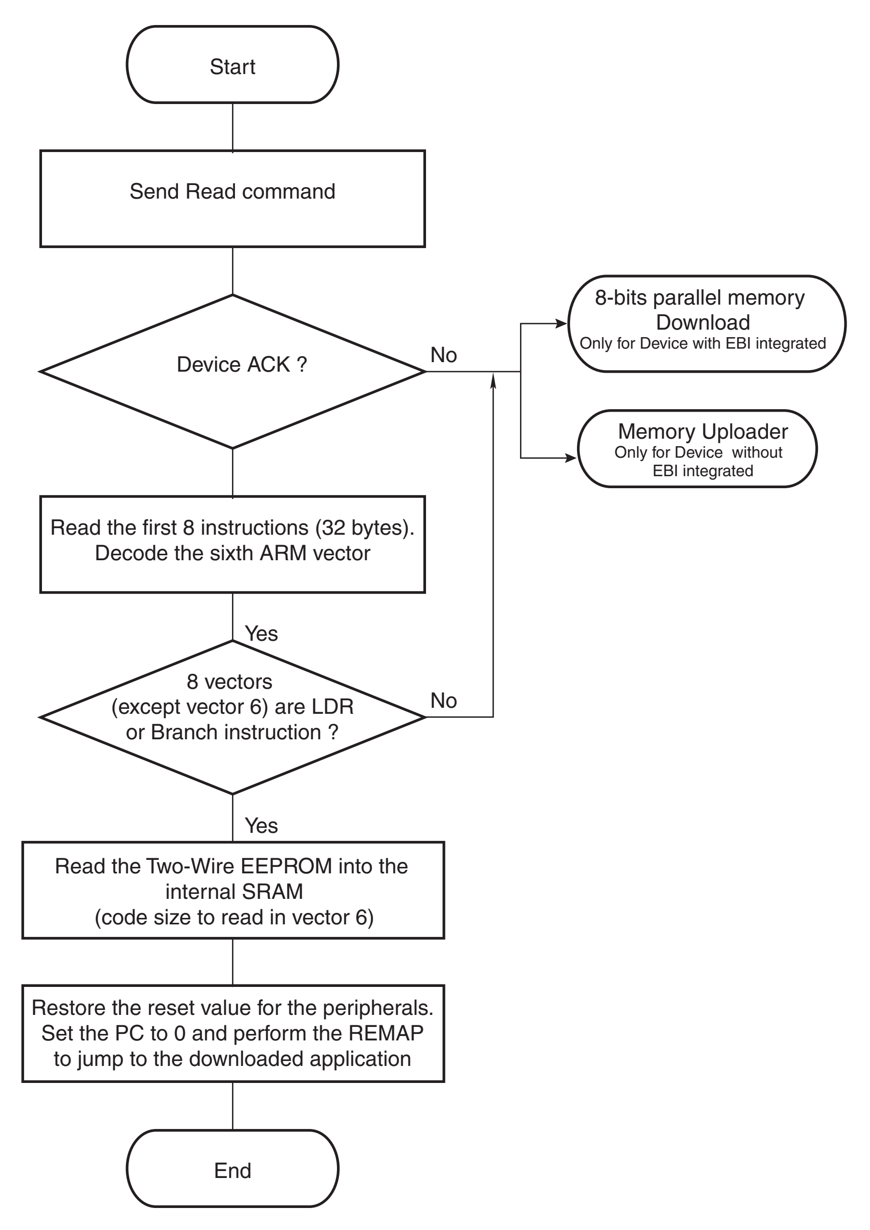
串行 DataFlash 下载

引导程序支持所有的 Atmel DataFlash 器件。 Table 21 总结了所有器件中包括 ARM 向量 6 在内的所有参数。

DataFlash 有一个状态寄存器，用来确定访问器件时所需的所有参数。

为与未来的 DataFlash 兼容，在 ARM 向量 6 中进行编码。

Figure 20. 串行 DataFlash 下载



串行两线 EEPROM 下载

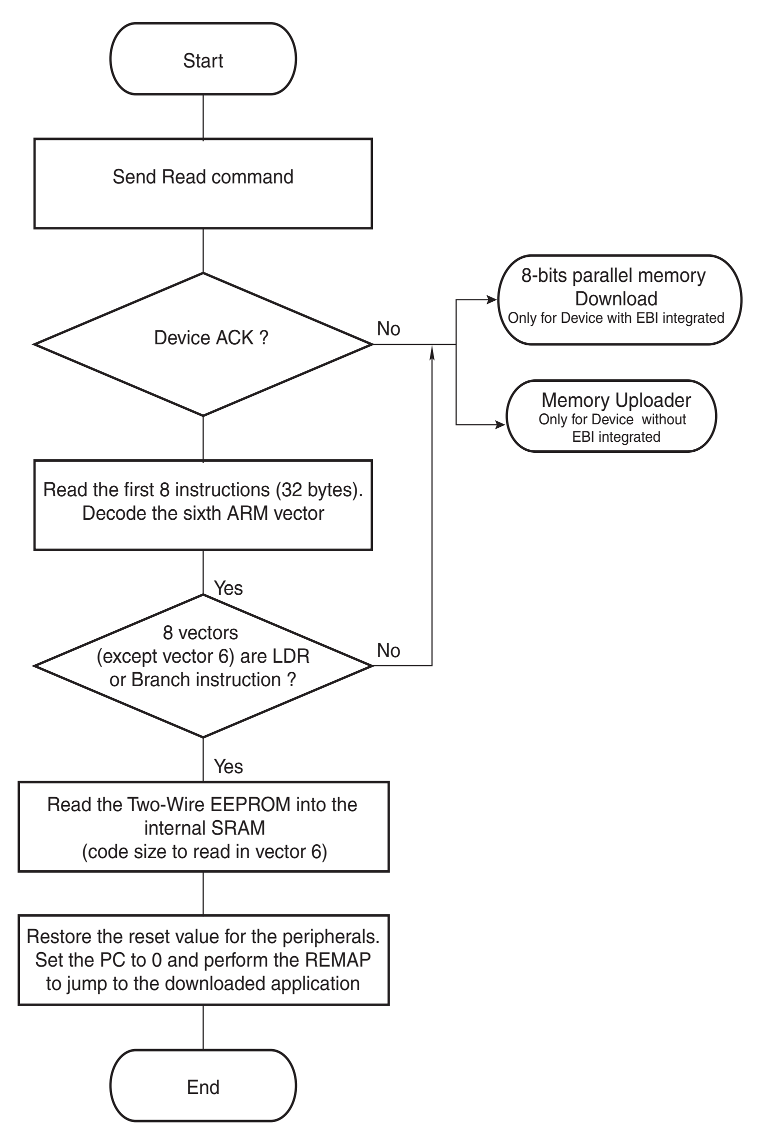
通常，串行 EEPROM 没有标识码。 bootloader 检查在第一次读时的应答。器件两线总线地址为 0x0。

bootloader 支持的器件见 Table 23。

Table 23. 支持的 EEPROM 器件

器件	大小	组织
AT24C16A	16 Kbits	16 字节页写
AT24C164	16 Kbits	16 字节页写
AT24C32	32 Kbits	32 字节页写
AT24C64	64 Kbits	32 字节页写
AT24C128	128 Kbits	64 字节页写
AT24C256	256 Kbits	64 字节页写
AT24C512	528 Kbits	128 字节页写

Figure 21. 串行两线 EEPROM 下载

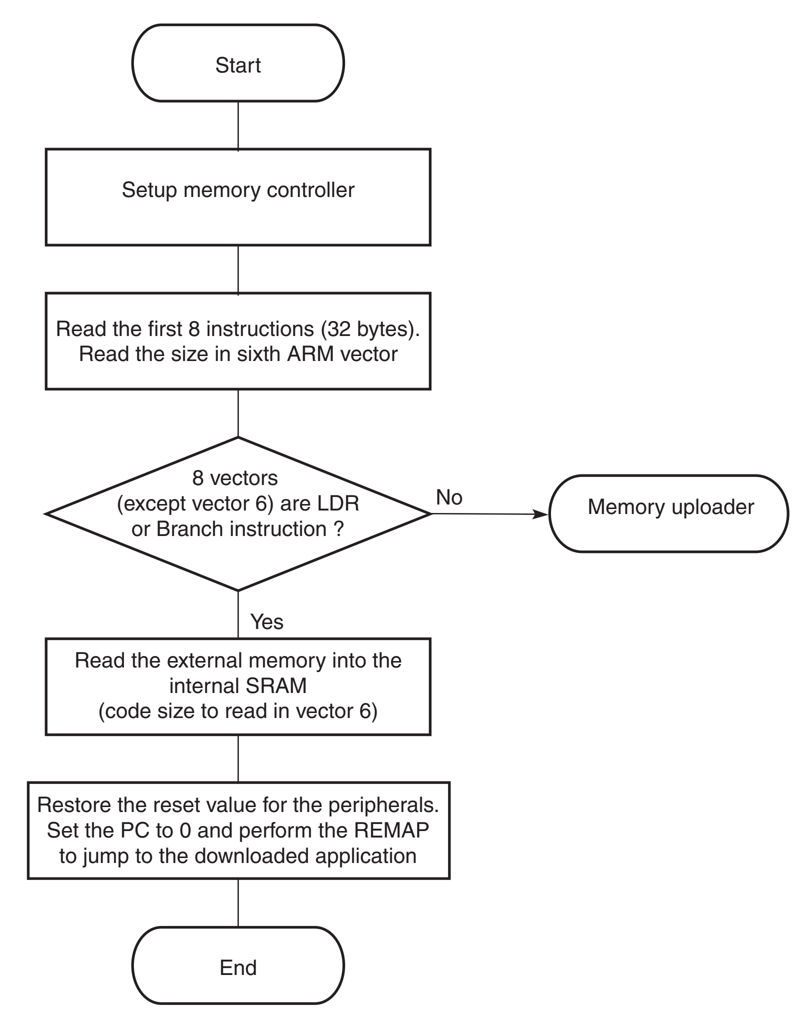


8位并行Flash下载(适用于含 EBI 器件)

若产品集成了外部总线接口 (EBI)，则支持 8 位并行 Flash 下载。

当 NCS0 配置为 8 位数据总线宽时， bootloader 支持 EBI 所支持的所有 8 位存储器器件。

Figure 22. 8 位并行 Flash 下载





Boot Uploader

若在Bootloader序列中未发现有效引导器件，将串行通信器件初始化(DBGU与USB器件端口)。

- 初始化 DBGU 串行端口 (115200 波特率， 8， N， 1) 并启动 Xmodem 协议
- 初始化 USB 器件端口并启动 DFU 协议
- 下载应用程序

引导 Uploader 执行 DFU 与 Xmodem 协议上传应用程序到内部 SRAM 的 0x0020_0000 地址。

引导程序为变量与堆栈在内部 SRAM 开辟一片存储区域。为防止上载错误，载入的应用程序至少要小于 SRAM 3K。

下载完成后，外设寄存器复位，中断禁用并重新映射。重新映射后，内部 SRAM 地址为 0x0000_0000，内部ROM地址为0x0010_0000。然后将PC设置为0。它于重映射前由管道中取指并在映射后执行。该取指周期对下载映射执行。

外部信道

DBGU 串行端口

上传是通过 DBGU 串行端口初始化为 115200 波特率， 8， n， 1 执行。

DBGU 发送字符 ‘C’ (0x43) 以启动 Xmodem 协议。任何执行该协议的终端可向目标发送应用文件。发送的二进制文件的大小由产品内置的 SRAM 大小决定 (参见微控制器资料以确定内置微控制器 SRAM 的大小)。由于 Xmodem 协议需要在 SRAM 存储器中运行，因此二进制文件大小必须小于 SRAM 大小。

Xmodem 协议

Xmodem 协议支持 128 字节长度块。该协议使用两个字符的 CRC-16 以保证最大位误差检测。

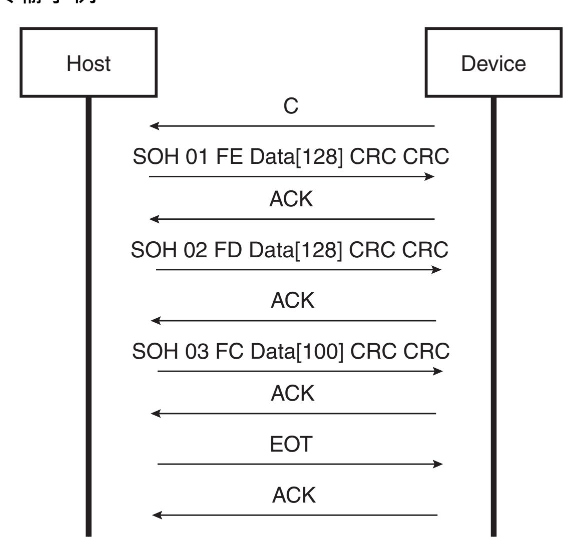
Xmodem 协议使用 CRC 以确保发送与接收成功传输报告。每个传输模块如下：

<SOH><blk #><255-blk #><--128 数据字节 --><checksum> 在：

- <SOH> = 01 hex
- <blk #> = 二进制数，从 01 开始，每次加 1，由 0FFH 转到 00H (而非 01)
- <255-blk #> = 1 的补码 blk#.
- <checksum> = 2 字节 CRC16

Figure 23 给出使用该协议传输图。

Figure 23. Xmodem 传输示例



USB 器件端口

使用 USB 端口需要 48 MHz USB 时钟。它在器件初始化配置 PLLB 前编程。

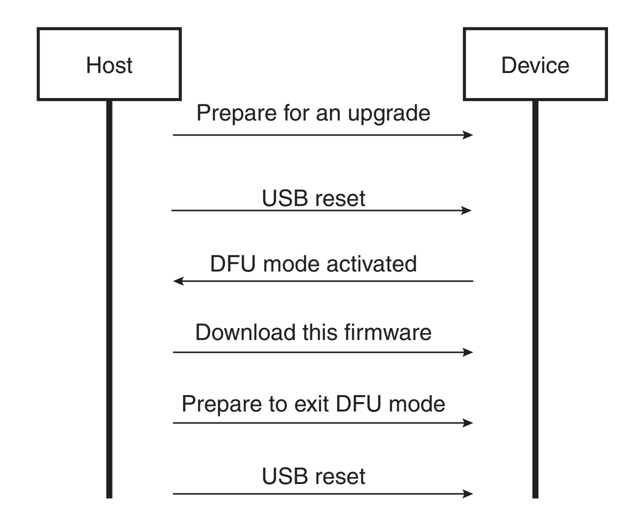
DFU 协议

DFU 允许升级 USB 器件固件。 DFU 算法是 USB 规范的一部分，详见 “USB Device Firmware Upgrade Specification, Rev. 1.0”。

进行固件升级有四个步骤：

1. 计数：器件获得主机容量。
2. 重新配置：主机与器件同意开始固件升级。
3. 传输：主机将固件映射传输到器件。主机与器件间出现状态请求以保持同步。
4. 表现：一旦器件向主机报告其已完成重编程操作，主机产生复位，器件执行更新固件。

Figure 24. 协议



软硬件限制

引导程序的软件限制为：

- 载入的代码大小要小于内置于产品中的 SRAM 大小。
- 在 TWI 总线上器件 EEPROM 地址必须为 0。
- 代码由器件地址 0x0000_0000(DataFlash， EEPROM) 下载到内部 SRAM 地址 0x0000_0000(重新映射后)。
- 下载代码必须位置独立或连接在地址 0x0000_0000 处。

引导程序硬件限制：

- DataFlash 必须连接在 SPI 的 NPCS0 上。
- 8 位并行 Flash 必须连接到 EBI 的 NCS0 上 (适用于集成了 EBI 的器件)。

SPI与TWI驱动器使用几个后备功能PIO与器件通信。当应用程序使用PIO时必须小心。引导期间连接的器件可能被无意驱动，可能在 SPI 或 TWI 的输出引脚与连接的器件间产生电路冲突。

为保证正确的操作，将受限器件插到其它引脚或通过设置 BMS 位启动外部 16 位并行存储器 (若产品集成 EBI)。

Table 24 包含在执行引导程序时驱动的引脚。若未发现正确的引导程序在 6 ms 一周期的引导序列中驱动这些引脚。通过 TWI 下载 64K 字节需 5 秒，因为 TWI 位速率为 100 Kbits/s。

DataFlash 由 12 MHz 的 SPCK 信号驱动，下载 64K 字节的时间缩短为 66 ms。

在执行跳转到内部 SRAM 应用程序前，引导程序使用的所有 PIO 与外设设置到其复位状态。

Table 24. 引导程序执行时的引脚驱动

所用引脚	SPI (Dataflash)	TWI (EEPROM)
MOSI(1)	O	X
SPCK(1)	O	X
NPCS0(1)	O	X
TWD(1)	X	I/O
TWCK(1)	X	O

Note: 1. 见 “Peripheral Multiplexing on PIO Lines” on page 18。

内置软件服务

概述

内置软件服务是驱动频繁执行任务的资源的独立软件组件，面向对象的软件为访问服务组建应用提供了方便。

AT91 服务有如下用途：

- 给出专门用于 AT91 器件的软件示例。
- 可用于 AT91 系列产品中。
- 为存储于 ROM 中的软件提供接口。

软件服务主要特性：

- ATPCS 适用
- AINSI/ISO 标准 C 适用
- 在 ARM/Thumb 交互工作中编译
- ROM 进入服务
- 提供 Tempo、 Xmodem 及 DataFlash 服务
- CRC 及正弦表

服务定义

服务结构

结构定义

服务结构定义于 C 头文件中。

结构由数据成员、函数指针 (方式) 组成，类似于类定义。对数据访问及程序访问没有保护。但是有些函数可由用户程序或服务使用，将其作为公用程序。同样，某些不能被调用的函数视为私有程序。这对数据同样适用。

程序

服务结构中，函数指针缺省初始化为标准函数。ROM 中仅有缺省标准函数。用户应用程序方式将使默认方式超载。

程序即不设定静态变量也不调用全局变量。所有服务结构程序调用是通过指针调用的。程序可不受限制的进行访问与更新服务数据。

同样的，程序中没有轮询。事实上，启动程序可启动函数 (对给定的事例读)、获取状态 (通过读取获得 ?) 并调回，通过启动程序初始化。因此，使用服务时客户程序通过启动读取与状态轮询实现同步读取；或当启动读操作时指定回叫异步读取。

服务入口

除 ***ROM Entry Service*** 外，每个 AT91 服务定义的函数名为 `AT91F_Open_<Service>`。这是为服务定义的唯一入口。即使函数 `AT91F_Open_<Service>` 可与对象构造函数类比，但它们在初始化服务结构未分配内存，因此无法作为构造器使用。因此用户应用程序必须给它分配空间。

示例

```
// 指定服务结构
AT91S_Pipe pipe;
// 开启服务
AT91PS_Pipe pPipe = AT91F_OpenPipe(&pipe, …);
```

AT91 服务定义中，服务结构程序指针初始化为默认程序。服务结构的其它区域初始化为默认值或按照函数 `AT91F_Open_<Service>` 的参数。



总的来说，应用程序必须知道服务结构与 AT91F_Open_<Service> 函数位置。

默认函数 AT91F_Open_<Service> 可由应用程序或含有应用程序定义功能的函数重新进行定义。

服务使用

开启服务

通过初始化服务结构确定入口。除 ROM 入口服务外，开启函数与每个服务结构有关。因此，用户端只有 AT91F_Open_<service> 函数可见。对服务程序的访问是通过服务结构函数指针实现。

函数 AT91F_Open_<service> 中至少有一个变量：服务结构指针必须分配到其它位置。它返回基服务结构或服务结构指针。

函数 AT91F_Open_<service> 初始化所有数据成员与程序指针。服务结构中所有函数指针指向服务函数。

该方法优点是向服务提供一个单入口。服务程序通过开启函数初始化且每个成员可以被覆盖。

覆盖程序

所有 ROM 中提供的服务定义为默认方式。这些程序可能不适应项目需求。通过项目程序定义可能覆盖默认程序。

方法是一个指针对应一个函数。该指针是通过 AT91F_Open_<Service> 函数进行初始化的。为在一个服务中覆盖一个或几个程序函数指针必须定义到新程序。

服务可覆盖一个程序，几个程序甚至所有程序。下面的例子中，服务功能是用户定义的，但仍工作在相同的数据结构中。

Note: 调用默认函数 AT91F_Open_<Service> 确保所有程序与数据初始化。

这可通过写一个新的 My_OpenService() 函数实现。新的开启函数必须调用库定义函数 AT91F_Open_<Service>，然后更新一个或几个函数指针：

Table 25. 通过覆盖开启服务函数来覆盖程序

<table>
<tr><th>ROM 中默认服务行为</th><th>通过 My_ChildMethod 覆盖 AT91F_ChildMethod</th></tr>
<tr><td><pre>
// Defined in embedded_services.h
typedef struct _AT91S_Service {
  char data;
  char (*MainMethod) ();
  char (*ChildMethod) ();
} AT91S_Service, * AT91PS_Service;

// Defined in obj_service.c (in ROM)
char AT91F_MainMethod ()
{
}

char AT91F_ChildMethod ()
{
}

// Init the service with default methods
AT91PS_Service AT91F_OpenService(
AT91PS_Service pService)
{
  pService->data = 0;
  pService->MainMethod =AT91F_MainMethod;
  pService->ChildMethod=AT91F_ChildMethod;
  return pService;
}
</pre></td><td><pre>
// My_ChildMethod will replace AT91F_ChildMethod
char My_ChildMethod ()
{
}

// Overloading Open Service Method
AT91PS_Service My_OpenService(
AT91PS_Service pService)
{
  AT91F_OpenService(pService);

// Overloading ChildMethod default value
  pService->ChildMethod= My_ChildMethod;
  return pService;
}

// Allocation of the service structure
AT91S_Service service;

// Opening of the service
AT91PS_Service pService =
My_OpenService(&service);
</pre></td></tr>
</table>





也可在使用 AT91F_Open_<Service> 程序后直接覆盖：

Table 26. 不覆盖开启服务函数来覆盖程序 .

ROM 中默认服务行为	通过 My_ChildMethod 覆盖 AT91F_ChildMethod

// Defined in embedded_services.h
typedef struct _AT91S_Service {
  char data;
  char (*MainMethod) ();
  char (*ChildMethod) ();
} AT91S_Service, * AT91PS_Service;
 	
// Defined in obj_service.c (in ROM)
char AT91F_MainMethod ()
{
}
 	
char AT91F_ChildMethod ()
{
}
// Init the service with default methods
AT91PS_Service AT91F_OpenService(
AT91PS_Service pService)
{
  pService->data = 0;
  pService->MainMethod =AT91F_MainMethod;
  pService->ChildMethod=AT91F_ChildMethod;
  return pService;
}
	// My_ChildMethod will replace AT91F_ChildMethod
	char My_ChildMethod ()
	{
	}
	
	// Allocation of the service structure
	AT91S_Service service;
	
	// Opening of the service
	AT91PS_Service pService =
	AT91F_OpenService(&service);
	
	// Overloading ChildMethod default value
	pService->ChildMethod= My_ChildMethod;

内置软件服务

定义

部分 AT91 内置 ROM。多数情况下， ROM 集成一个 bootloader 与几个服务以加速应用程序减少应用代码大小。

当软件置于 ROM 时，每个对象地址 (函数、常量、表格等) 必须与用户应用程序相关。通过向连接器提供一个地址列表来实现。对每个版本的 ROM 都要提供一个新的地址表且所有客户应用程序必须重新编译。

内置软件服务提供其它方式访问存于 ROM 中的对象。对于每个内置服务 , 用户应用程序只需要服务入口地址。

即使这些服务只有一个入口 (AT91F_Open_<Service> 函数)，它们必须列入连接器。内置软件服务通过提供一个专门服务：ROM 入口服务来解决这个问题。

该服务目的是实现只需一个地址即可访问整个 ROM。

ROM 入口服务

产品 ROM 入口服务是命为 AT91S_RomBoot 的结构。该结构中的某些成员指向存储于 ROM 中的所有服务的开放功能 (AT91F_Open_<Service> 函数)，但要 CRC 与 Sine 排列。因此，只公布 AT91S_RomBoot 地址。

Table 27. ROM 入口服务初始化与使用开启服务程序

<table>
<tr><th>应用存储空间</th><th>ROM 存储空间</th></tr>
<tr><td><pre>
// Init the ROM Entry Service
AT91S_RomBoot const *pAT91;
pAT91 = AT91C_ROM_BOOT_ADDRESS;

// Allocation of the service structure
AT91S_CtlTempo tempo;

// Call the Service Open method
pAT91->OpenCtlTempo(&tempo, ...);

// Use of tempo methods
tempo.CtlTempoCreate(&tempo, ...);
</pre></td><td><pre>
AT91S_TempoStatus AT91F_OpenCtlTempo(
    AT91PS_CtlTempo pCtlTempo,
    void const *pTempoTimer)
{

...
}

AT91S_TempoStatus AT91F_CtlTempoCreate (
    AT91PS_CtlTempo pCtrl,
    AT91PS_SvcTempo pTempo)
{
...

}
</pre></td></tr>
</table>

应用程序获取 ROM 入口服务地址并初始化 AT91S_RomBoot 结构的对象实例。为获取存于 ROM 中的其它服务的开启服务程序，应用程序使用 AT91S_RomBoot 中适当的的成员。

异常向量后，在 ROM 起始位置可找到 AT91S_RomBoot 地址。

Tempo 服务

简介

Tempo 服务允许单硬件系统定时器同时支持几个软件定时器运行。作为对象事件通告程序。

定义了两个对象用于控制 Tempo 服务 : AT91S_CtlTempo 与 AT91S_SvcTempo。

应用程序声明一个与硬件系统定时器相关的 AT91S_CtlTempo，此外应用程序还控制 AT91S_SvcTempo 的一列实例。

当应用程序需要其它定时器时 , 它要求 AT91S_CtlTempo 创建一个 AT91S_SvcTempo 实例 , 之后应用程序初始化 AT91S_SvcTempo 所有设置。

Tempo 服务描述

Table 28. Tempo 服务程序

默认使用的相关函数指针与程序	描述

`// Typical Use:`
`pAT91->`**`OpenCtlTempo`**`(...);`
 	
`// Default Method:`
`AT91S_TempoStatus AT91F_OpenCtlTempo(`
`AT91PS_CtlTempo pCtlTempo,`
`void const *pTempoTimer)`
	`AT91S_RomBoot` 成员结构。
	Tempo 服务中相应的开启服务。
	输入参数：
	控制 Tempo 对象指针。
	系统定时器说明结构指针。
	输出参数：
	若 OpenCtrlTempo 成功返回 0 ；否则返回 1。

`// Typical Use:`
`AT91S_CtlTempo ctlTempo;`
`ctlTempo.`**`CtlTempoStart`**`(...);`
 	
`// Default Method:`
`AT91S_TempoStatus AT91F_STStart(void * pTimer)`
	`AT91S_CtlTempo` 结构成员。
	启动相关的硬件系统定时器。
	输入参数：
	指向对应系统定时器说明符结构的有效参数。
	输出参数：
	返回 2。

`// Typical Use:`
`AT91S_CtlTempo ctlTempo;`
`ctlTempo.`**`CtlTempoIsStart`**`(...);`
 	
`// Default Method:`
`AT91S_TempoStatus AT91F_STIsStart(`
`AT91PS_CtlTempo pCtrl)`
	`AT91S_CtlTempo` 结构成员
	输入参数：
	控制 Tempo 对象指针。
	输出参数：
	返回系统定时器状态寄存器

`// Typical Use:`
`AT91S_CtlTempo ctlTempo;`
`ctlTempo.`**`CtlTempoCreate`**`(...);`
 	
`// Default Method:`
`AT91S_TempoStatus AT91F_CtlTempoCreate (`
`AT91PS_CtlTempo pCtrl,`
`AT91PS_SvcTempo pTempo)`
	`AT91S_CtlTempo` 结构成员
	`AT91S_SvcTempo` 列表中插入软件定时器。
	输入参数：
	控制 Tempo 对象指针。
	插入控制 Tempo 对象指针。
	输出参数：
	若软件创建 tempo 返回 0 ；否则返回 1。

`// Typical Use:`
`AT91S_CtlTempo ctlTempo;`
`ctlTempo.`**`CtlTempoRemove`**`(...);`
 	
`// Default Method:`
`AT91S_TempoStatus AT91F_CtlTempoRemove`
`(AT91PS_CtlTempo pCtrl,`
`AT91PS_SvcTempo pTempo)`
	`AT91S_CtlTempo` 结构成员。
	删除列表中的软件定时器。
	输入参数：
	控制 Tempo 对象指针。
	删除控制 Tempo 对象指针。
	输出参数：
	创建 tempo 返回 0 ；否则返回 1。

Table 28. Tempo 服务程序 (Continued)

默认使用的相关函数指针与程序	描述

`// Typical Use:`
`AT91S_CtlTempo ctlTempo;`
`ctlTempo.`**`CtlTempoTick`**`(...);`
 	
`// Default Method:`
`AT91S_TempoStatus AT91F_CtlTempoTick (AT91PS_CtlTempo pCtrl)`
	`AT91S_CtlTempo` 结构成员。
	刷新列表中软件定时器。若运行回叫则更新其暂停及校验。因此该功能可使用在例如当使用周期定时器时硬件定时器启动新的周期中断。
	输入参数 :
	控制 Tempo 对象指针。
	输出参数 :
	返回 1。

`// Typical Use:`
`AT91S_SvcTempo svcTempo;`
`svcTempo.`**`Start`**`(...);`
 	
`// Default Method:`
`AT91S_TempoStatus AT91F_SvcTempoStart (`
`AT91PS_SvcTempo pSvc,`
`unsigned int timeout,`
`unsigned int reload,`
`void (*callback) (AT91S_TempoStatus, void *),`
`void *pData)`
	`AT91S_SvcTempo` 结构成员。
	启动软件定时器。
	输入参数 :
	服务 Tempo 对象指针。
	申请暂停。
	定期在暂停结束后重载 tempo 次数。
	暂停结束运行程序回叫。
	允许在当前服务有异常分支。
	输出参数 :
	返回 1。

`// Typical Use:`
`AT91S_SvcTempo svcTempo;`
`svcTempo.`**`Stop`**`(...);`
 	
`// Default Method:`
`AT91S_TempoStatus AT91F_SvcTempoStop (`
`AT91PS_SvcTempo pSvc)`
	`AT91S_SvcTempo` 结构成员。
	强制停止软件定时器。
	输入参数 :
	服务 Tempo 对象指针。
	输出参数 :
	返回 1。

Note: `AT91S_TempoStatus` 只适用于无符号整型数。



使用服务

第一步使用 ROM 入口服务获取开启服务程序 `AT91F_OpenCtlTempo` 地址。

在应用存储器空间分配 `AT91S_CtlTempo` 与 `AT91S_SvcTempo` 示例：

```
// 分配服务与控制 tempo
AT91S_CtlTempo ctlTempo;
AT91S_SvcTempo svcTempo1;
```

通过调用 `AT91F_OpenCtlTempo` 函数初始化 `AT91S_CtlTempo` 示例：

```
// 初始化服务
pAT91->OpenCtlTempo(&ctlTempo, (void *) &(pAT91->SYSTIMER_DESC));
```

此时，应用程序可使用 `AT91S_CtlTempo` 服务成员。

若应用程序希望覆盖对象成员，现在就可以。例如，若 `AT91F_CtlTempoCreate(&ctlTempo, &svcTempo1)` 程序由应用程序定义的 `my_CtlTempoCreate(...)` 取代，其步骤如下：

```
// 覆盖 AT91F_CtlTempoCreate
ctlTempo.CtlTempoCreate = my_CtlTempoCreate;
```

多数情况下通过调用 `AT91S_CtlTempo` 服务的 `AT91F_CtlTempoCreate` 程序来初始化 `AT91S_SvcTempo` 对象：

```
// 初始化与 AT91S_CtlTempo 对象连接的 svcTempo1
ctlTempo.CtlTempoCreate(&ctlTempo, &svcTempo1);
```

通过调用 svcTempo1 对象启动程序来启动暂停。回叫是在倒数完成还是读 svcTempo1 对象的 `TickTempo` 成员检测到暂停状态时启动是由函数参数决定的。

```
// 启动暂停
svcTempo1.Start(&svcTempo1,100,0,NULL,NULL);
// Wait for the timeout of 100 (unity depends on the timer programmation)
// No repetition and no callback.
while (svcTempo1.TickTempo);
```

当应用程序需要其它软件定时器控制暂停时：

- 在应用存储器空间分配 `AT91S_SvcTempo` 示例。

```
// 分配服务
AT91S_SvcTempo svcTempo2;
```

- 调用 `AT91S_CtlTempo` 服务的 `AT91F_CtlTempoCreate` 程序初始化 `AT91S_SvcTempo` 对象：

```
// 初始化与 AT91S_CtlTempo 对象连接的 svcTempo2
ctlTempo.CtlTempoCreate(&ctlTempo, &svcTempo2);
```

Xmodem 服务

简介

Xmodem 服务是通信管道抽象层的一个应用程序。该层是独立于媒体 (USART、USB 等) 的 ，给在抽象媒介上的读与写提供一个入口点。

通信管道服务

管道通信结构是一个虚拟结构，包含了与通信媒介和存储管理无关的读写缓冲器所需的所有函数。

管道结构定义：

- 通信服务结构 `AT91PS_SvcComm` 的指针
- 缓冲管理结构 `AT91PS_Buffer` 的指针
- 读写函数指针
- 与读写函数相关的回叫函数指针

下列结构定义管道对象：

```
typedef struct _AT91S_Pipe
{
  // 与外设及缓冲器的管道链接
  AT91PS_SvcComm pSvcComm;
  AT91PS_Buffer  pBuffer;

  // 含变量的回叫函数
  void (*WriteCallback) (AT91S_PipeStatus, void *);
  void (*ReadCallback)  (AT91S_PipeStatus, void *);
  void *pPrivateReadData;
  void *pPrivateWriteData;

  // 管道函数
  AT91S_PipeStatus (*Write) (
    struct _AT91S_Pipe   *pPipe,
    char const *         pData,
    unsigned int         size,
    void                 (*callback) (AT91S_PipeStatus, void *),
    void                 *privateData);
  AT91S_PipeStatus (*Read) (
    struct _AT91S_Pipe  *pPipe,
    char                *pData,
    unsigned int        size,
    void                (*callback) (AT91S_PipeStatus, void *),
    void                *privateData);
  AT91S_PipeStatus (*AbortWrite) (struct _AT91S_Pipe  *pPipe);
  AT91S_PipeStatus (*AbortRead) (struct _AT91S_Pipe *pPipe);
  AT91S_PipeStatus (*Reset) (struct _AT91S_Pipe *pPipe);
  char (*IsWritten) (struct _AT91S_Pipe *pPipe,char const *pVoid);
  char (*IsReceived) (struct _AT91S_Pipe *pPipe,char const *pVoid);
} AT91S_Pipe, *AT91PS_Pipe;
```

Xmodem 协议给出如何使用通信管道的示例。

缓冲器结构说明

AT91PS_Buffer 是 AT91S_Buffer 结构管理缓冲器的指针。该结构内置如下函数：

- 管理读缓冲器的函数指针
- 管理写缓冲器的函数指针

为适用缓冲器管理所有函数均可被覆盖。

引导 ROM 源代码中为 ***Xmodem*** 服务提供了一个简单可执行的缓冲器管理。

```
typedef struct _AT91S_Buffer
{
```



```
  struct _AT91S_Pipe *pPipe;
  void *pChild;

  // 通过管道调用函数
  AT91S_BufferStatus (*SetRdBuffer)     (struct _AT91S_Buffer *pSBuffer, char
*pBuffer, unsigned int Size);
  AT91S_BufferStatus (*SetWrBuffer)     (struct _AT91S_Buffer *pSBuffer, char const
*pBuffer, unsigned int Size);
  AT91S_BufferStatus (*RstRdBuffer)     (struct _AT91S_Buffer *pSBuffer);
  AT91S_BufferStatus (*RstWrBuffer)     (struct _AT91S_Buffer *pSBuffer);
  char (*MsgWritten)      (struct _AT91S_Buffer *pSBuffer, char const *pBuffer);
  char (*MsgRead)         (struct _AT91S_Buffer *pSBuffer, char const *pBuffer);

  // 通过外设调用函数
  AT91S_BufferStatus (*GetWrBuffer)     (struct _AT91S_Buffer *pSBuffer, char const
**pData, unsigned int *pSize);
  AT91S_BufferStatus (*GetRdBuffer)     (struct _AT91S_Buffer *pSBuffer, char
**pData, unsigned int *pSize);
  AT91S_BufferStatus (*EmptyWrBuffer)   (struct _AT91S_Buffer *pSBuffer, unsigned
int size);
  AT91S_BufferStatus (*FillRdBuffer)    (struct _AT91S_Buffer *pSBuffer, unsigned
int size);
  char (*IsWrEmpty)      (struct _AT91S_Buffer *pSBuffer);
  char (*IsRdFull)       (struct _AT91S_Buffer *pSBuffer);
} AT91S_Buffer, *AT91PS_Buffer;
```

SvcComm 结构说明

SvcComm 结构提供低层函数与管道对象间的接口。

它包含有初始化低层函数的函数指针 (如 SvcXmodem)。

使用 SvcComm 的 Xmodem 服务执行示例。

```
typedef struct _AT91S_Service
{
  // 程序：
  AT91S_SvcCommStatus (*Reset)  (struct _AT91S_Service *pService);
  AT91S_SvcCommStatus (*StartTx)(struct _AT91S_Service *pService);
  AT91S_SvcCommStatus (*StartRx)(struct _AT91S_Service *pService);
  AT91S_SvcCommStatus (*StopTx) (struct _AT91S_Service *pService);
  AT91S_SvcCommStatus (*StopRx) (struct _AT91S_Service *pService);
  char                (*TxReady)(struct _AT91S_Service *pService);
  char                (*RxReady)(struct _AT91S_Service *pService);
  // 数据：
  struct _AT91S_Buffer *pBuffer; // Link to a buffer object
  void *pChild;
} AT91S_SvcComm, *AT91PS_SvcComm;
```





SvcXmodem 结构说明

SvcXmodem 服务在 Xmodem 协议中可重复执行。它只支持 128 字节的分组格式并提供读写函数。 SvcXmodem 结构定义如下 :

- 读或写处理程序初始化指针
- 处理 xmodem 包 crc 函数指针
- 校验包头函数指针
- 数据校验函数指针

该结构下 , Xmodem 协议可在所有媒介中使用 (USART、 USB 等). 只有私有程序可能需要覆盖以使 Xmodem 协议适用新媒介。

Xmodem 默认使用 USART 发送与接收包。读写函数由外设数据控制器执行以减少中断开销。假设 USART 已初始化 , 分配存储缓冲器并对中断编程。

需要服务周期定时器来管理暂停并定时发送字符 “C” (参见 Xmodem 协议)。该特性由 Tempo 服务提供。

下面结构定义 Xmodem 服务 :

```
typedef struct _AT91PS_SvcXmodem {

  // 公共程序 :
  AT91S_SvcCommStatus (*Handler) (struct _AT91PS_SvcXmodem *, unsigned int);
  AT91S_SvcCommStatus (*StartTx) (struct _AT91PS_SvcXmodem *, unsigned int);
  AT91S_SvcCommStatus (*StopTx)  (struct _AT91PS_SvcXmodem *, unsigned int);

  // 私有程序 :
  AT91S_SvcCommStatus (*ReadHandler)  (struct _AT91PS_SvcXmodem *, unsigned int
csr);
  AT91S_SvcCommStatus (*WriteHandler) (struct _AT91PS_SvcXmodem *, unsigned int
csr);
  unsigned short      (*GetCrc)       (char *ptr, unsigned int count);
  char                (*CheckHeader)  (unsigned char currentPacket, char *packet);
  char                (*CheckData)    (struct _AT91PS_SvcXmodem *);

  AT91S_SvcComm  parent;      // 基类
  AT91PS_USART pUsart;

  AT91S_SvcTempo tempo; // 链接 AT91S_Tempo 对象

  char          *pData;
  unsigned int  dataSize;        // = XMODEM_DATA_STX 或 XMODEM_DATA_SOH
  char          packetDesc[AT91C_XMODEM_PACKET_SIZE];
  unsigned char packetId;         // 当前包
  char          packetStatus;
  char          isPacketDesc;
  char          eot;            // 传输结束
} AT91S_SvcXmodem, *AT91PS_SvcXmodem
```

Xmodem 服务说明

Table 29. Xmodem 服务程序

默认的相关函数指针与程序	说明

`// Typical Use:`
`pAT91->`**`OpenSvcXmodem`**`(...);`
 	
`// Default Method:`
`AT91PS_SvcComm AT91F_OpenSvcXmodem(`
`  AT91PS_SvcXmodem pSvcXmodem,`
`  AT91PS_USART pUsart,`
`  AT91PS_CtlTempo  pCtlTempo)`
	`AT91S_RomBoot` 结构成员。
	Xmodem 服务相应的开启服务。
	
	输入参数：
	SvcXmodem 结构指针。
	USART 结构指针。
	CtlTempo 结构指针。
	输出参数：
	返回 Xmodem 服务指针结构。

`// Typical Use:`
`AT91S_SvcXmodem svcXmodem;`
`svcXmodem.`**`Handler`**`(...);`
 	
`// Default read handler:`
`AT91S_SvcCommStatus`
`AT91F_SvcXmodemReadHandler(AT91PS_SvcXmodem`
`pSvcXmodem, unsigned int csr)`
 	
`// Default write handler:`
`AT91S_SvcCommStatus`
`AT91F_SvcXmodemWriteHandler(AT91PS_SvcXmodem`
`pSvcXmodem, unsigned int csr)`
	`AT91S_SvcXmodem` 结构成员。
	xmodem 读写功能中断处理。
	输入参数：
	Xmodem 服务结构指针。
	csr：usart 通道状态寄存器。
	输出参数：
	xmodem 读写状态。



使用服务

下面给出初始化与使用 Xmodem 服务步骤：

```
Variables definitions:
AT91S_RomBoot const *pAT91; // 结构包含开启服务功能
AT91S_SBuffer   sXmBuffer; // Xmodem 缓冲器分配
AT91S_SvcXmodem svcXmodem; // Xmodem 服务结构分配
AT91S_Pipe      xmodemPipe;// xmodem 管道通信结构
AT91S_CtlTempo  ctlTempo; // Tempo 结构
AT91PS_Buffer pXmBuffer; // 缓冲器结构指针
AT91PS_SvcComm pSvcXmodem; // 媒介结构指针
Initialisations
// Call Open methods:
pAT91 = AT91C_ROM_BOOT_ADDRESS;
// OpenCtlTempo on the system timer
pAT91->OpenCtlTempo(&ctlTempo, (void *) &(pAT91->SYSTIMER_DESC));
ctlTempo.CtlTempoStart((void *) &(pAT91->SYSTIMER_DESC));
// Xmodem buffer initialisation
pXmBuffer     = pAT91->OpenSBuffer(&sXmBuffer);
pSvcXmodem = pAT91->OpenSvcXmodem(&svcXmodem, AT91C_BASE_DBGU, &ctlTempo);
// Open communication pipe on the xmodem service
pAT91->OpenPipe(&xmodemPipe, pSvcXmodem, pXmBuffer);
// Init the DBGU peripheral
// Open PIO for DBGU
  AT91F_DBGU_CfgPIO();
  // Configure DBGU
  AT91F_US_Configure (
    (AT91PS_USART) AT91C_BASE_DBGU,     // DBGU 基地址
    MCK,              // Master Clock
    AT91C_US_ASYNC_MODE,        // 模式寄存器编程
    BAUDRATE ,               // 波特率编程
    0);                    // 时间保护编程
  // Enable Transmitter
  AT91F_US_EnableTx((AT91PS_USART) AT91C_BASE_DBGU);
  // Enable Receiver
  AT91F_US_EnableRx((AT91PS_USART) AT91C_BASE_DBGU);
// Initialize the Interrupt for System Timer and DBGU （中断共享）
// Initialize the Interrupt Source 1 for SysTimer and DBGU
  AT91F_AIC_ConfigureIt(AT91C_BASE_AIC,
           AT91C_ID_SYS,
           AT91C_AIC_PRIOR_HIGHEST,
           AT91C_AIC_SRCTYPE_INT_LEVEL_SENSITIVE,
           AT91F_ASM_ST_DBGU_Handler);

  // Enable SysTimer and DBGU interrupt
  AT91F_AIC_EnableIt(AT91C_BASE_AIC, AT91C_ID_SYS);


xmodemPipe.Read(&xmodemPipe, (char *) BASE_LOAD_ADDRESS, MEMORY_SIZE,
XmodemProtocol, (void *) BASE_LOAD_ADDRESS);
```

DataFlash 服务

简介

DataFlash服务允许串行外设(SPI)支持对几个串行DataFlash与DataFlash卡的读取、编程与擦除操作。

该服务基于 SPI 中断，由指定的处理程序管理。它还使用相应的 PDC 寄存器。

DataFlash 服务中命令有效信息见 DataFlash 相关文件。

DataFlash 服务说明

Table 30. DataFlash 服务程序

默认的相关函数指针与程序	说明

`// Typical Use:`
`pAT91->`**`OpenSvcDataFlash`**`(...);`
 	
`// Default Method:`
`AT91PS_SvcDataFlash AT91F_OpenSvcDataFlash (`
`const AT91PS_PMC pApmc,`
`AT91PS_SvcDataFlash pSvcDataFlash)`
	`AT91S_RomBoot` 结构成员。
	DataFlash 服务中相应的开启服务程序。
	输入参数 :
	PMC 寄存器说明结构指针。
	DataFlash 服务结构指针。
	输出参数 :
	返回 DataFlash 服务指针结构。

`// Typical Use:`
`AT91S_SvcDataFlash svcDataFlash;`
`svcDataFlash.`**`Handler`**`(...);`
 	
`// Default Method:`
`void AT91F_DataFlashHandler(`
`AT91PS_SvcDataFlash pSvcDataFlash,`
`unsigned int status)`
	`AT91S_SvcDataFlash` 成员结构。
	SPI 固定外设 C 中断处理程序。
	输入参数 :
	DataFlash 服务结构指针。
	状态：相应的中断检测与 SPI 有效 (通过 SPI 状态寄存器将 SPI 状态寄存器屏蔽)。
	必须将中断处理程序推入 SPI。
	输出参数 :
	无。

`// Typical Use:`
`AT91S_SvcDataFlash svcDataFlash;`
`svcDataFlash.`**`Status`**`(...);`
 	
`// Default Method:`
`AT91S_SvcDataFlashStatus`
`AT91F_DataFlashGetStatus(AT91PS_DataflashDesc pDesc)`
	`AT91S_SvcDataFlash` 成员结构。
	读 DataFlash 状态寄存器。
	输入参数 :
	DataFlash 说明结构 (服务结构成员) 指针。
	输出参数 :
	若 DataFlash 忙返回 0。
	若 DataFlash 就绪返回 1。

`// Typical Use:`
`AT91S_SvcDataFlash svcDataFlash;`
`svcDataFlash.`**`AbortCommand`**`(...);`
 	
`// Default Method:`
`void`
`AT91F_DataFlashAbortCommand(AT91PS_DataflashDesc pDesc)`
	`AT91S_SvcDataFlash` 成员结构。
	允许复位 PDC 及中断
	输入参数 :
	DataFlash 说明结构 (服务结构成员) 指针。
	输出参数 :
	无。



Table 30. DataFlash 服务程序

默认的相关函数指针与程序	说明

// Typical Use:
AT91S_SvcDataFlash svcDataFlash;
svcDataFlash.**PageRead**(...);
 	
// Default Method:
AT91S_SvcDataFlashStatus AT91F_DataFlashPageRead (
AT91PS_SvcDataFlash pSvcDataFlash,
unsigned int src,
unsigned char *dataBuffer,
int sizeToRead)
	AT91S_SvcDataFlash 成员结构。
	读 DataFlash 中页。
	输入参数 :
	DataFlash 服务结构指针。
	DataFlash 地址。
	数据缓冲器目的指针。
	读取字节数。
	输出参数 :
	若 DataFlash 忙返回 0。
	若 DataFlash 就绪返回 1。

// Typical Use:
AT91S_SvcDataFlash svcDataFlash;
svcDataFlash.**ContinuousRead**(...);
 	
// Default Method:
AT91S_SvcDataFlashStatus
AT91F_DataFlashContinuousRead (
AT91PS_SvcDataFlash pSvcDataFlash,
int src,
unsigned char *dataBuffer,
int sizeToRead)
	AT91S_SvcDataFlash 成员结构。
	继续流读取。
	输入参数 :
	DataFlash 服务结构指针。
	DataFlash 地址。
	数据缓冲器目的指针。
	读取字节数。
	输出参数 :
	若 DataFlash 忙返回 0。
	若 DataFlash 就绪返回 1。

// Typical Use:
AT91S_SvcDataFlash svcDataFlash;
svcDataFlash.**ReadBuffer**(...);
 	
// Default Method:
AT91S_SvcDataFlashStatus AT91F_DataFlashReadBuffer
(
AT91PS_SvcDataFlash pSvcDataFlash,
unsigned char BufferCommand,
unsigned int bufferAddress,
unsigned char *dataBuffer,
int sizeToRead)
	AT91S_SvcDataFlash 成员结构。
	读内部 DataFlash SRAM 缓冲器 1 或 2。
	输入参数 :
	DataFlash 服务结构指针。
	选择内部 DataFlash 缓冲器 1 或 2 命令。
	DataFlash 地址。
	数据缓冲器目的指针。
	读取字节数。
	输出参数 :
	若 DataFlash 忙返回 0。
	若 DataFlash 就绪返回 1。
	若 DataFlash 错误命令返回 4。
	若 DataFlash 错误地址返回 5。

// Typical Use:
AT91S_SvcDataFlash svcDataFlash;
svcDataFlash.**MainMemoryToBufferTransfert**(...);
 	
// Default Method:
AT91S_SvcDataFlashStatus
AT91F_MainMemoryToBufferTransfert(
AT91PS_SvcDataFlash pSvcDataFlash,
unsigned char BufferCommand,
unsigned int page)
	AT91S_SvcDataFlash 成员结构。
	读内部 SRAM 缓冲器 1 或 2 页。
	输入参数 :
	DataFlash 服务结构指针。
	选择内部 DataFlash 缓冲器 1 或 2 命令。
	读取页。
	输出参数 :
	若 DataFlash 忙返回 0。
	若 DataFlash 就绪返回 1。
	若 DataFlash 错误命令返回 4。

Table 30. DataFlash 服务程序

默认的相关函数指针与程序	说明

// Typical Use:
AT91S_SvcDataFlash svcDataFlash;
svcDataFlash.**PagePgmBuf**(...);
 	
// Default Method:
AT91S_SvcDataFlashStatus AT91F_DataFlashPagePgmBuf(
AT91PS_SvcDataFlash pSvcDataFlash,
unsigned char BufferCommand,
unsigned char *src,
unsigned int dest,
unsigned int SizeToWrite)
	AT91S_SvcDataFlash 成员结构。
	通过内部 SRAM 缓冲器 1 或 2 页编程。
	输入参数：
	DataFlash 服务结构指针。
	选择内部 DataFlash 缓冲器 1 或 2 命令。
	源缓冲器。
	DataFlash 目的地址。
	写入字节数。
	输出参数：
	若 DataFlash 忙返回 0。
	若 DataFlash 就绪返回 1。
	若 DataFlash 错误命令返回 4。

// Typical Use:
AT91S_SvcDataFlash svcDataFlash;
svcDataFlash.**WriteBuffer**(...);
 	
// Default Method:
AT91S_SvcDataFlashStatus AT91F_DataFlashWriteBuffer (
AT91PS_SvcDataFlash pSvcDataFlash,
unsigned char BufferCommand,
unsigned char *dataBuffer,
unsigned int bufferAddress,
int SizeToWrite)
	AT91S_SvcDataFlash 成员结构。
	写数据到内部 SRAM 缓冲器 1 或 2。
	输入参数：
	DataFlash 服务结构指针。
	选择内部 DataFlash 缓冲器 1 或 2 命令。
	数据缓冲写指针。
	内部缓冲器地址。
	写入字节数。
	输出参数：
	若 DataFlash 忙返回 0。
	若 DataFlash 就绪返回 1。
	若 DataFlash 错误命令返回 4。
	若 DataFlash 错误地址返回 5。

// Typical Use:
AT91S_SvcDataFlash svcDataFlash;
svcDataFlash.**WriteBufferToMain**(...);
 	
// Default Method:
AT91S_SvcDataFlashStatus AT91F_WriteBufferToMain (
AT91PS_SvcDataFlash pSvcDataFlash,
unsigned char BufferCommand,
unsigned int dest)
	AT91S_SvcDataFlash 成员结构。
	写内部缓冲器到 DataFlash 主存储器。
	输入参数：
	DataFlash 服务结构指针。
	选择内部 DataFlash 缓冲器 1 或 2 命令。
	DataFlash 主存储器地址。
	输出参数：
	若 DataFlash 忙返回 0。
	若 DataFlash 就绪返回 1。



Table 30. DataFlash 服务程序

默认的相关函数指针与程序	说明

`// Typical Use:`
`AT91S_SvcDataFlash svcDataFlash;`
`svcDataFlash.`**`PageErase`**`(...);`
 	
`// Default Method:`
`AT91S_SvcDataFlashStatus AT91F_PageErase (`
`AT91PS_SvcDataFlash pSvcDataFlash,`
`unsigned int PageNumber)`
	`AT91S_SvcDataFlash` 成员结构。
	擦除 DataFlash 中页。
	输入参数：
	服务 DataFlash 对象指针。
	页擦除。
	输出参数：
	若 DataFlash 忙返回 0。
	若 DataFlash 就绪返回 1。

`// Typical Use:`
`AT91S_SvcDataFlash svcDataFlash;`
`svcDataFlash.`**`BlockErase`**`(...);`
 	
`// Default Method:`
`AT91S_SvcDataFlashStatus AT91F_BlockErase (`
`AT91PS_SvcDataFlash pSvcDataFlash,`
`unsigned int BlockNumber )`
	`AT91S_SvcDataFlash` 成员结构。
	擦除 8 页中的一块。
	输入参数：
	服务 DataFlash 对象指针。
	块擦除。
	输出参数：
	若 DataFlash 忙返回 0。
	若 DataFlash 就绪返回 1。

`// Typical Use:`
`AT91S_SvcDataFlash svcDataFlash;`
`svcDataFlash.`**`MainMemoryToBufferCompare`**`(...);`
 	
`// Default Method:`
`AT91S_SvcDataFlashStatus`
`AT91F_MainMemoryToBufferCompare(`
`AT91PS_SvcDataFlash pSvcDataFlash,`
`unsigned char BufferCommand,`
`unsigned int page)`
	`AT91S_SvcDataFlash` 成员结构。
	比较页与一个内部 SRAM 缓冲器内容。
	输入参数：
	服务 DataFlash 对象指针。
	内部 SRAM DataFlash 缓冲器比较命令。
	页比较。
	输出参数：
	若 DataFlash 忙返回 0。
	若 DataFlash 就绪返回 1。
	若 DataFlash 错误命令返回 4。

Note: AT91S_SvcDataFlashStatus 只适用于无符号整型数。

使用服务

第一步使用 ROM 入口服务查找 `AT91F_OpenSvcDataFlash` 开启服务程序地址。

1. 在应用存储空间给 `AT91S_SvcDataFlash` 与 `AT91S_Dataflash` 分配空间：

```
// 分配服务与器件结构。
AT91S_SvcDataFlash svcDataFlash;
AT91S_Dataflash Device; // AT91S_SvcDataFlash 服务成员
```

   然后通过调用 `AT91F_OpenSvcDataFlash` 函数初始化 `AT91S_SvcDataFlash`：

```
// 初始化服务
pAT91->OpenSvcDataFlash (AT91C_BASE_PMC, &svcDataFlash);
```

2. 初始化 `SPI` 中断：

```
// 初始化 SPI 中断
at91_irq_open ( AT91C_BASE_AIC,AT91C_ID_SPI,3,
                    AT91C_AIC_SRCTYPE_INT_LEVEL_SENSITIVE ,AT91F_spi_asm_handler);
```

3. 用正确特性配置 DataFlash 结构并与 `AT91S_SvcDataFlash` 服务结构中的器件结构连接：

```
// ATMEL AT45DB321B DataFlash 示例
Device.pages_number = 8192;
Device.pages_size = 528;
Device.page_offset = 10;
Device.byte_mask = 0x300;
// 与服务结构连接
svcDataFlash.pDevice = &Device;
```

4. 现在可使用不同程序。下面是对 528 字节一页的读取示例：

```
// RxBufferDataFlash 中读操作结果
unsigned char RxBufferDataFlash[528];
svcDataFlash.PageRead(&svcDataFlash,
        (50*528),RxBufferDataFlash,528);
```



CRC 服务

简介

该服务不同于处理类型服务，其结构不同。它由一个阵列及一些通过 `AT91S_RomBoot` 结构直接访问的程序。

CRC 服务说明

Table 31. CRC 服务说明

程序与阵列可用	说明

`// Typical Use:`
`pAT91->`**`CRC32`**`(...);`
 	
`// Default Method:`
`void CalculateCrc32(`
`const unsigned char *address,`
`unsigned int size,`
`unsigned int *crc)`
	该函数提供对字节数据驱动 32 位 CRC 表。该 CRC 称为 CCITT CRC32。
	输入参数：
	数据缓冲器指针。
	缓冲器大小。
	CRC 结果指针。
	输出参数：
	无。

`// Typical Use:`
`pAT91->`**`CRC16`**`(...);`
 	
`// Default Method:`
`void CalculateCrc16(`
`const unsigned char *address,`
`unsigned int size,`
`unsigned short *crc)`
	该函数提供对字节数据驱动 16 位 CRC 表。 CRC 由 POLYNOME 0x8005 计算。
	输入参数：
	数据缓冲器指针。
	缓冲器大小。
	CRC 结果指针。
	输出参数：
	无

`// Typical Use:`
`pAT91->`**`CRCHDLC`**`(...);`
 	
`// Default Method:`
`void CalculateCrcHdlc(`
`const unsigned char *address,`
`unsigned int size,`
`unsigned short *crc)`
	该函数提供对字节数据驱动 16 位 CRC 表。该 CRC 称为 HDLC CRC。
	输入参数：
	数据缓冲器指针。
	缓冲器大小。
	CRC 结果指针。
	输出参数：
	无。

`// Typical Use:`
`pAT91->`**`CRCCCITT`**`(...);`
 	
`// Default Method:`
`void CalculateCrc16ccitt(`
`const unsigned char *address,`
`unsigned int size,`
`unsigned short *crc)`
	该函数提供对字节数据驱动 16 位 CRC 表。该 CRC 称为 CCITT CRC16 (POLYNOME = 0x1021).
	输入参数：
	数据缓冲器指针。
	缓冲器大小。
	CRC 结果指针。
	输出参数：
	无

`// Typical Use:`
`char reverse_byte;`
`reverse_byte = pAT91->`**`Bit_Reverse_Array`**`[...];`
 	
`// Array Embedded:`
`const unsigned char bit_rev[256]`
	位翻转阵列：允许翻转一个八位字节。经常用于数学算法中。
	
	用于 CRC16 计算。

使用服务

计算 256 字节缓冲器的 CRC16 CCITT 并将其存入 crc16 变量：

```
// 计算 CRC16 CCITT
```

```
unsigned char BufferToCompute[256];
short crc16;
... (BufferToCompute Treatment)
pAT91->CRCCCITT(&BufferToCompute,256,&crc16);
```

Sine 服务

简介

该服务不同于处理类型服务，其结构不同。它由一个阵列及一些通过 `AT91S_RomBoot` 结构直接访问的程序。

Sine 服务说明

Table 32. Sine 服务说明

程序与阵列可用	说明

`// Typical Use:`
`pAT91->`**`Sine`**`(...);`
 	
`// Default Method:`
`short AT91F_Sinus(int step)`
	该函数返回给定的正弦波形 16 位振幅码。
	输入参数：
	正弦步长。对应于放大计算精度。由使用的正弦阵列决定。此处，阵列 256 值对应 180 度。
	输出参数：
	正弦波形振幅。

`// Typical Use:`
`short sinus;`
`sinus = pAT91->`**`SineTab`**`[...];`
 	
`// Array Embedded:`
`const short AT91C_SINUS180_TAB[256]`
	精度为 256 值对应 180 度的正弦阵列。

AT91RM9200 复位控制器

概述

该章说明 AT91RM9200 复位信号及对其正确使用以保证器件正确操作。

AT91RM9200 有两个独立的复位输入线：NRST 与 NTRST。

- 用户接口寄存器初始化 (定义每个外设的用户接口) 并：
  - 对所需 bootup 信号采样；
  - 强迫处理器在地址零取指。
- 初始化内置 ICE TAP 控制器。

NRST 信号必须看作系统复位信号而且用户设计逻辑驱动复位信号时必须小心。NTRST 主要使用于使用内部仿真器单元的硬件调试接口，而且对其初始化不影响 ARM® 处理器正常工作。该线也可由板上逻辑驱动。

NRST 与 NTRST 均为低电平有效，在 AT91RM92000 异步复位逻辑。

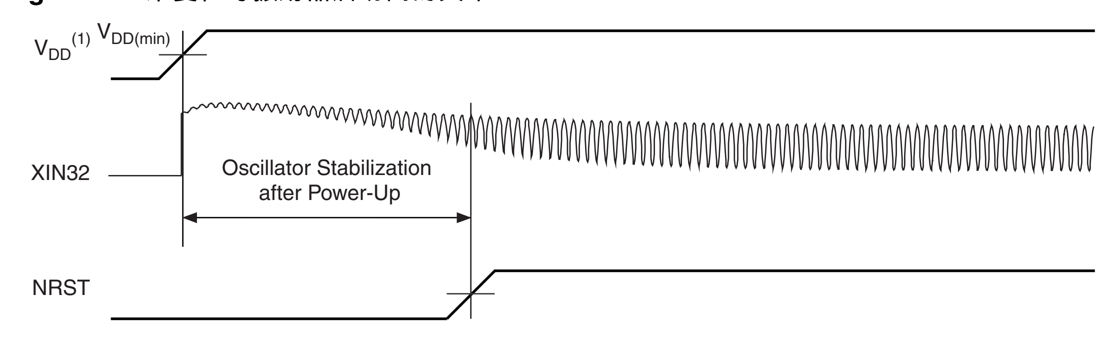
复位条件

NRST 条件

NRST是低电平激活。系统上电后，AT91RM9200必须执行一个上电复位(也称为 “冷”复位)。在过渡状态下，它强制保持复位信号为低直到电源达到工作标称电压而且振荡器工作频率稳定为止。一般当认为电环境不在标称值时提供门限电压限制。不只上电，休眠与掉电时同样会出现 NRST 信号。门限电压的选择是根据 AT91RM9200 最小工作电压决定的，在 Figure 25 中为 VDD(See “DC Characteristics” on page 596.)。

复位延迟的选择是由低频率振荡器启动时间确定的，如 Figure 25 所示 (见 “32 kHz Oscillator Characteristics” on page 599)。

Figure 25. 冷复位与振荡器启动间的关系



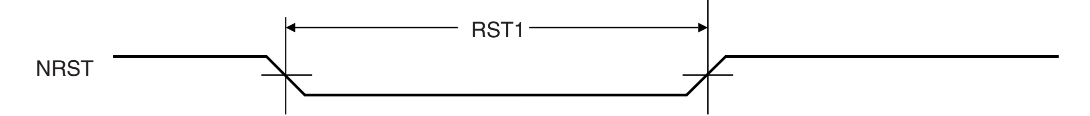
Note: 1. VDD 适用于 VDDIOM、 VDDIOP、 VDDPLL、 VDDOSC 及 VDDCORE。

NRST 也可在除上电序列外的其它序列中出现，例如一个手动命令。插入可异步执行，但是由复位退出时必须与默认的激活时钟同步。在正常操作期间， NRST 有效须保持一个最小的延迟时间以保证工作正确，见 Figure 26 与 Table 33。

Table 33. 最小复位脉冲宽度

符号	参数	最小脉宽	单位
RST1	NRST 最小脉宽	92	µs

Figure 26. NRST 判决



NTRST 判决

与 NRST 信号一样，上电时 NTRST 信号在电源电压未达到最小推荐工作电压时必须保持有效 (见 See “DC Characteristics” on page 596.)。 TCK 上时钟无需确认该复位请求。

与 NRST 信号一样， NTRST 也可通过如手动命令或 ICE 接口动作插入除上电序列外的其它序列中。插入与去除可异步执行，但激活时必须有一个最小延迟时间 (见 “JTAG/ICE Timings” on page 621)。

复位管理

系统复位

NRST 信号提供系统复位功能。

复位信号用以强制微控制器单元出现初始状态：

- 对引导模式选择 (BMS) 逻辑状态采样。
- 保存用户接口默认状态 (默认值)。
- 请求处理器由地址零读取下一条指令。

处理器寄存器只定义了程序计数器与当前程序状态寄存器。当微控制器的 NRST 输入插入，处理器立即停止当前执行的指令。

系统复位电路必须考虑两种复位请求类型：

- 上电序列中的冷复位；
- 用户复位请求。

两者效果一样，但视 NRST 引脚有不同的插入时间。事实上，冷复位必须覆盖系统启动时间。用户复位请求的延迟时间相对较短。

测试复位

NTRST 信号提供测试功能。

NTRST 控制引脚对选定的 TAP 控制器初始化。复位时的 TAP 控制器是由最后有效的 NRST 中 JTAGSEL 引脚提供的初始逻辑状态决定的。

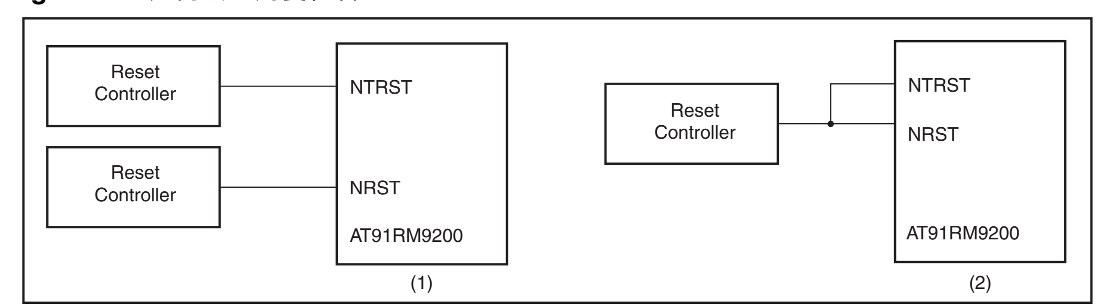
边界扫描模式中，在插入 NTRST 后，IDCODE 指令设置到测试 - 逻辑 - 复位控制器状态的输出指令寄存器中。

此外， ICE 模式下复位步骤如下：

- 由调试模式推出内核。
- 需要 IDCORE 指令。

如 Figure 27 所示，边界扫描与 ICE 模式中的复位，根据系统复位是上电还是用户请求，可在相同或不同的电路中执行。

Figure 27. 分离或通用复位管理



Notes: 1. 开发时 NRST 与 NTRST 处理是在调试模式下。
2. 成品后 NRST 与 NTRST 处理。

为在开发时调试阶段 NRST 与 NTRST 的分离中获益更多 , 用户必须单独管理 Figure 27 所示示例 (1) 的信号 ; 一旦调试完成 , 生产时两信号可轻易整合起来管理 , 如 Figure 27 所示示例 (2)。

复位控制器需求特性

下表给出为获得 AT91RM9200 处理器的最优系统，复位控制器的需求。

Table 34. 复位控制器功能小结

特性	说明
电源监控	覆盖系统上电、休眠及掉电的过渡状态。
复位有效暂停周期	通过延时期间保持复位信号覆盖引导振荡器的启动时间。
手动复位命令	由逻辑命令插入复位信号并保持复位信号较复位有效暂停周期为短。

存储控制器 (MC)

概述

存储控制器 (MC) 管理 ASB 总线并最多达 4 个主机的访问控制。它通过一个总线判决器和一个地址译码器将 4G 字节的地址空间分区来访问内置的 SRAM 与 ROM ,内置外设及通过外部总线接口 (EBI) 的外部存储器 。它还可通过一个异常中止状态与一个失调检波器来帮助应用程序调试。

存储控制器允许从内置 ROM 或连接在 EBI 片选 0 上的外部非易失性存储器上引导。重映射在命令在内置 SRAM 上切换 ARM 向量 (0x0 - 0x20) 地址。

RM9200 存储控制器的主要性能如下:

- 可编程总线判决器处理 4 个主机
  - 内部总线由 ARM920T、 PDC、 USB 主机端口及以太网 MAC 主机共享
  - 主机可配置为 0 ~ 7 中任意一个优先级
- 地址译码器提供的选择:
  - 8 个外部 256-M 字节存储空间
  - 4 个内部 1-M 字节存储空间
  - 1 个 256-M 字节内置外设空间
- 引导模式选项
  - 内部或外部非易失性引导存储器
  - 通过 BMS 引脚在复位时的采样值来选择
- 异常中断状态寄存器
  - 引起异常中断的源。类型及访问的所有参数全部保存
- 失调检测器
  - 所有数据访问的校准检查
  - 失调引起的异常中断
- 重映射命令
  - 提供位于引导 NVM 上的内部 SRAM 的重映射



方框图

Figure 28. 存储控制器方框图

功能说明

存储控制器 (MC) 处理内部 ASB 总线及最多达 4 个的主机访问判决。

它由以下部分组成：

- 一个总线判决器
- 一个地址译码器
- 一个异常中断状态
- 一个失调检测器

存储控制器只处理低位在前模式的访问 。所有主机必须工作在低位在前模式中。

总线判决器

存储控制器有一个用户可编程的总线判决器。主机可配置为 0 ~ 7 中任意一个优先级，其中 7 级为最高级。总线判决器在寄存器 MC_MPR (主机优先级寄存器) 中编程。

不同的主机优先级可以相同。若两个有相同优先级的主机请求出现 ，总线判决器将使用如下的缺省优先级来决定谁先服务：主机 0，主机 1，主机 2，主机 3。

主机为：

- ARM920T 为主机 0
- 外设数据控制器为主机 1
- USB 主机端口作为主机 2
- 以太网 MAC 作为主机 3

地址译码器

存储控制器通过地址译码器来对 32 位地址总线的最高 4 位进行译码并定义 11 个独立区域：

- 256-M 字节的内部存储器地址空间
- 8 个 256-M 字节地址空间，每个都分别分配给外部总线接口的 8 个片选口中的一个。
- 1 个 256-M 字节的内置外设地址空间
- 1536M 字节在访问时返回异常中断的未定义地址空间

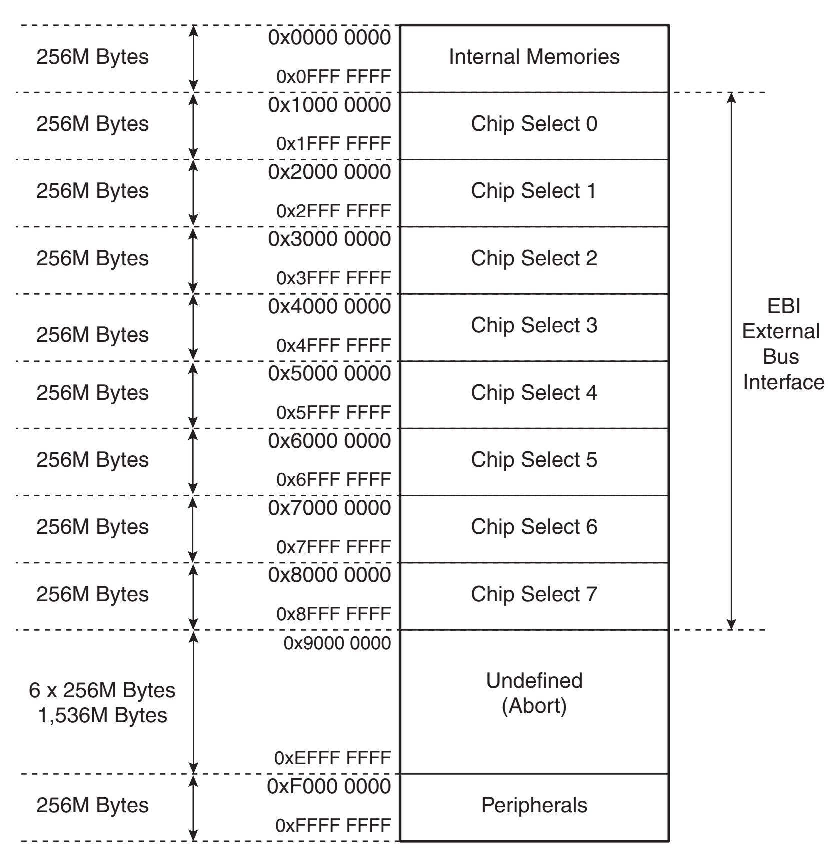




外部存储空间

Figure 29 给出 256-M 字节存储空间的分配。

Figure 29. 外部存储空间



内部存储器映射

在内部存储地址空间中，存储控制器的地址译码器译码 8 比特地址位为内置存储分配 1-M 字节空间。

分配的存储域可访问 1-M 字节地址空间，故可在地址范围内 n 次访问， n 为 1M 除以存储器大小的值。

当访问内部存储器的未定义域地址时，即地址超出 0x0040 0000，地址译码器向主机返回一个异常中断。

Figure 30. 重新映射后的内部存储器映射图

Address	Area	Size
0x0000 0000 – 0x000F FFFF	Internal Memory Area 0	1M Byte
0x0010 0000 – 0x001F FFFF	Internal Memory Area 1 Internal ROM	1M Byte
0x0020 0000 – 0x002F FFFF	Internal Memory Area 2 Internal SRAM	1M Byte
0x0030 0000 – 0x003F FFFF	Internal Memory Area 3 USB Host Port	1M Byte
0x0040 0000 – 0x0FFF FFFF	Undefined Area (Abort)	252M bytes

256M bytes

内部存储器域 0

取决于 BMS 引脚状态并作为重新映射命令函数，在地址 0x0 上的存储器映射是不同的。在执行片上 ROM 重新映射命令 (BMS = 1) 前或非易失性存储器连接到外部芯片 0(BMS = 0) 中，映射到内部存储器域 0。执行重映射命令后，内部 SRAM 地址 0x0020 0000 映射到内部存储器域 0。映射到内部存储器域 0 的存储器可通过其原始地址或在地址 0x0 访问。

内部存储器域 0 的头 32 字节包含 ARM 处理器异常向量。

Table 35. 由 BMS 与重映射命令决定的内部存储器域

BMS 状态	重映射前		重映射后
	1	**0**	**X**
内部存储器域 0	内部 ROM	外部存储器域 0	内部 SRAM

引导模式选择

BMS 引脚状态允许器件由内部或外部外的其它引导存储器引导，事实上，在复位前的最后两个时钟周期中 BMS 引脚输入电平情况确定所选择的引导存储器类型：

- 高电平，内部 ROM 在内部存储器域 1 内映射，通过内部存储器域 0 访问。
- 低电平，外部存储器域 0，从地址 0x1000 0000 访问，也可通过内部存储器域 0 访问。

BMS 引脚可复用为 I/O 口。复位后，该引脚可作为标准 PIO 口来使用。

重映射命令

执行重映射命令后，内部 SRAM 通过内部存储器域 0 来访问。

由于 ARM 向量 (复位、异常中断、数据异常中断、预取异常中断、未定义指令、中断及快速中断) 映射到地址 0x0 ～ 0x20，重映射命令允许用户通过软件控制重新动态定义这些向量。

重映射命令可通过向 MC_RCR (重映射控制寄存器) RCB 域写 1 使用存储控制器用户接口来进行访问。

向 MC_RCR RCB 域写 1 可取消重映射命令，其实质上是一个切换命令 。它提供了一个简单的方法，将芯片配置为与复位后相同，使得对用户定义的引导序列调试变的简单。

Table 35 on page 125 概括了两种通信对于映射到地址 0x0 上存储器的影响。

异常中断状态

异常中断出现有两个原因：

- 对未定义地址进行访问
- 对失调地址进行访问

当异常中断出现后，不管是哪个访问发生异常，将向所有主机发送信号。但只有引起异常的主机处理该信号。

异常中断信号导致在 ARM9TDMI 产生一个异常。注意，从处理器的角度来看，异常中断也能由 ARM920T 的存储器管理单元产生，但这显然不是存储控制器所管理的，因此此处不做讨论。

外设数据控制器不处理异常中断输入信号（这就是为什么在 Figure 28 中没有连接）。UHP 向 Hc 中断状态寄存器发出无法恢复报告并复位该操作。EMAC 通过其状态寄存器的 ABT 位向用户报告异常中断，这可能会产生一个中断。

存储控制器集成一个异常中断状态寄存器集以方便操作系统的调试或错误分析。

异常中断地址状态寄存器 (MC_AASR) 保存整个 32 位宽的异常中断地址。访问参数保存在异常中断状态寄存器 (MC_ASR) 中，其中包括：

- 请求空间大小 (ABTSZ 域)
- 访问类型，是数据读、写还是取代码 (ABTTYP 域)
- 访问是应对未定义地址访问 (UNDADD 位) 造成还是由于地址失调 (MISADD 位) 造成
- 引起异常中断的源 (MST0、 MST1、 MST2 及 MST3 位)
- 除在 MST 位跟踪的主机外，读寄存器 (SVMST0、SVMST1、SVMST2 及 SVMST3 位) 后其它主机是否出现异常

若处理器发生数据异常中断，保存数据访问地址。这对于要求分解指令和处理器前后状态关系的查找发生异常的地址将非常有用。

但在预取指异常时，由于预取指异常出现在 ARM 处理器流水中，因此地址可能改变。ARM 处理器仅在读指令执行后且可能此时发生了多个异常时才考虑预取指异常。所以在这种情况下，更好的方法是使用 ARM 处理器的异常连接寄存器中的内容。

失调检验器

存储控制器的失调检验器检测访问的连续性。

无论哪个主机，必须校验每次访问的访问大小及访问总线位 0 与位 1。若访问类型为字 (32 位) 且位 0 与位 1 非 0，或访问类型为半字 (16 位) 且位 0 与位 1 非 0，将向主机返回异常信号并取消此次访问。注意在取指时对 ARM 处理器访问不进行校验。

失调主要是错误的指针处理导致的软件错误引起的，这些错误通过对调试相位的检测是很难发现的。

由于请求地址保存于处理器异常状态寄存器而产生失调的指针地址保存在异常连接寄存器中，因此对于此类软件错误的检测与修正非常容易。

存储控制器中断

存储控制器本身不会产生中断。但如 Figure 28 所示，若 SDRAM 控制器检测到刷新错误，存储控制器由外部总线接口接收的中断信号可能激活。该中断信号通过存储控制器传输，不可以使能 / 禁用或返回其动作。

该存储控制器中断信号是 ORed，其它系统外设中断口 (RTC、 ST、 DBGU、 PMC) 在增强中断控制器源 1 中提供系统中断。

用户接口

基地址：0xFFFFFF00

Table 36. RM9200 存储控制器存储器映射

偏移	寄存器	名称	访问	复位状态
0x00	MC 重映射控制寄存器	MC_RCR	只写	
0x04	MC 异常状态寄存器	MC_ASR	只读	0x0
0x08	MC 异常地址状态寄存器	MC_AASR	只读	0x0
0x0C	MC 主机优先级寄存器	MC_MPR	读 / 写	0x3210
0x10 - 0x5C	保留位			
0x60	EBI 配置寄存器	见 EBI 数据手册		



MC 重映射控制寄存器

寄存器名称：MC_RCR

访问类型： 只写

绝对地址：0xFFFF FF00

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	RCB

- **RCB: 重映射命令位**

0：无效

1：该命令位是切换的：写 1 将选择取消或保存页零存储器器件的重映射。

MC 异常状态寄存器

寄存器名称： MC_ASR

访问类型： 只读

复位值： 0x0

绝对地址： 0xFFFF FF04

31	30	29	28	27	26	25	24
–	–	–	–	SVMST3	SVMST2	SVMST1	SVMST0

23	22	21	20	19	18	17	16
–	–	–	–	MST3	MST2	MST1	MST0

15	14	13	12	11	10	9	8
–	–	–	–	ABTTYP		ABTSZ	

7	6	5	4	3	2	1	0
–	–	–	–	–	–	MISADD	UNDADD

- **UNDADD: 未定义地址异常状态**

0：最新的异常不是由于对未定义地址的地址空间访问所引起的。

1：最新的异常是由于对未定义地址的地址空间访问所引起的。

- **MISADD: 失调地址异常状态**

0：最新的异常不是由于地址失调所引起的。

1：最新的异常是由于地址失调所引起的。

- **ABTSZ: 异常大小状态**

ABTSZ		异常大小
0	0	字节
0	1	半字
1	0	字
1	1	保留

- **ABTTYP: 异常类型状态**

ABTTYP		异常类型
0	0	读数据
0	1	写数据
1	0	取代码
1	1	保留

- **MST0: ARM920T 异常源**

0：最新的异常访问不是由于 ARM920T 所造成的。

1：最新的异常访问是由于 ARM920T 所造成的。

- **MST1: PDC 异常源**

0：最新的异常访问不是由于 PDC 所造成的。

1：最新的异常访问是由于 PDC 所造成的。



- **MST2: UHP 异常源**

0：最新的异常访问不是由于 UHP 所造成的。

1：最新的异常访问是由于 UHP 所造成的。

- **MST3: EMAC 异常源**

0：最新的异常访问不是由于 EMAC 所造成的。

1：最新的异常访问是由于 EMAC 所造成的。

- **SVMST0: 保存 ARM920T 异常源**

0：自最后读 MC_ASR 未出现 ARM920T 引起的异常或通知 MST0 位。

1：自最后读 MC_ASR 至少出现一次 ARM920T 引起的异常。

- **SVMST1: 保存 PDC 异常源**

0：自最后读 MC_ASR 未出现 PDC 引起的异常或通知 MST1 位。

1：自最后读 MC_ASR 至少出现一次 PDC 引起的异常。

- **SVMST2: 保存 UHP 异常源**

0：自最后读 MC_ASR 未出现 UHP 引起的异常或通知 MST2 位。

1：自最后读 MC_ASR 至少出现一次 UHP 引起的异常。

- **SVMST3: 保存 EMAC 异常源**

0：自最后读 MC_ASR 未出现 EMAC 引起的异常或通知 MST3 位。

1：自最后读 MC_ASR 至少出现一次 EMAC 引起的异常。

MC 异常地址状态寄存器

寄存器名称： MC_AASR

访问类型：只读

复位值： 0x0

绝对地址： 0xFFFF FF08

31	30	29	28	27	26	25	24
ABTADD							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
ABTADD							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
ABTADD							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ABTADD							

- **ABTADD: 异常地址**

该域包含最新异常访问的地址。

MC 主机优先级寄存器

寄存器名称： MC_MPR

访问类型： 读 / 写

复位值： 0x3210

绝对地址：0xFFFF FF0C

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	MSTP3			–	MSTP2		

7	6	5	4	3	2	1	0
–	MSTP1			–	MSTP0		

- **MSTP0: ARM920T 优先级**
- **MSTP1: PDC 优先级**
- **MSTP2: UHP 优先级**
- **MSTP3: EMAC 优先级**

000：优先级最低

111：优先级最高

若优先级相同，主机 0 优先级最高，主机 3 优先级最低。

外部总线接口 (EBI)

概述

外部总线接口 (EBI) 设计用以确保多个外设和基于 ARM® 器件的内置控制存储器间的正确数据传输。静态存储器、SDRAM 及 Burst Flash 控制器均可作为 EBI 上的外部存储控制器。这些外部存储控制器可以处理多种类型的外部存储器以及外部设备，如 SRAM、 PROM、 EPROM、EEPROM、 Flash、 SDRAM 及 Burst Flash。

EBI通过集成电路支持CompactFlash与SmartMedia协议从而极大的降低了对外部组件的需求。此外， EBI 可处理多达 8 个外设的数据传输，每个外设分配 8 个在内置存储控制器中定义的地址空间。数据通过 16 位或 32 位数据总线进行传输，地址总线高达 26 位， 8 个芯片选择口 (NCS[7:0]) 和在不同外部存储控制器间复用的多个控制引脚进行。

EBI 特性：

- 集成了 3 个外部存储控制器：
  - 静态存储控制器
  - SDRAM 控制器
  - Burst Flash 控制器
- SmartMedia™ 和 CompactFlash™ 额外的逻辑支持
- 优化的外部总线：
  - 16 和 32 位数据总线
  - 多达 26 位的地址总线，可对 64-M 字节空间寻址
  - 8 个片选信号线，每个对应到 8 个存储域中的一个
  - 优化的引脚复用以减少外部存储器的等待时间
- 片选分配设定：
  - NCS0 上为 Burst Flash 控制器或静态存储控制器
  - NCS1 上为 SDRAM 控制器或静态存储控制器
  - NCS3 上为静态存储控制器，可选支持 SmartMedia
  - NCS4 - NCS6 上为静态存储控制器，可选支持 CompactFlash
  - NCS7 上为静态存储控制器



方框图

Figure 31 给出外部总线接口结构。

Figure 31. 外部总线接口结构

Memory Controller
ASB
Address Decoder
External Bus Interface
SDRAM Controller
Burst Flash Controller
Static Memory Controller
NAND Flash/ SmartMedia Logic
CompactFlash Logic
Chip Select Assignor
User Interface
MUX Logic
PIO
APB
D[15:0]
A0/NBS0
A1/NWR2/NBS2
A[15:2], A[22:18]
A16/BA0
A17/BA1
NCS0/BFCS
NCS1/SDCS
NCS2
NCS3/SMCS
NRD/NOE/CFOE
NWR0/NWE/CFWE
NWR1/NBS1/CFIOR
NWR3/NBS3/CFIOW
SDCK
SDCKE
RAS
CAS
SDWE
SDA10
D[31:16]
A[24:23]
A25/CFRNW
NCS4/CFCS
NCS5/CFCE1
NCS6/CFCE2
NCS7
BFCK
BFAVD
BFBAA/SMWE
BFOE
BFRDY/SMOE
BFWE
NWAIT

I/O 口线说明

Table 37. I/O 口线说明

名称	功能	类型	工作电平
	EBI		
D[31:0]	数据总线	I/O	
A[25:0]	地址总线	输出	
	SMC		
NCS[7:0]	片选线	输出	低
NWR[1:0]	写信号	输出	低
NOE	输出使能	输出	低
NRD	读信号	输出	低
NBS1	NUB：高字节选择	输出	低
NBS0	NLB：低字节选择	输出	低
NWE	写使能	输出	低
	CompactFlash 支持的 EBI		
CFCE[2:1]	CompactFlash 芯片使能	输出	低
CFOE	CompactFlash 输出使能	输出	低
CFWE	CompactFlash 写使能	输出	低
CFIOR	CompactFlash I/O 读信号	输出	低
CFIOW	CompactFlash I/O 写信号	输出	
CFRNW	CompactFlash 读非写信号	输出	低
CFCS	CompactFlash 片选线	输出	低
NWAIT	CompactFlash 等待信号	输入	低
	SmartMedia 支持的 EBI		
SMCS	SmartMedia 片选线	输出	低
SMOE	SmartMedia 输出使能	输出	低
SMWE	SmartMedia 写使能	输出	低
	SDRAM 控制器		
SDCK	SDRAM 时钟	输出	
SDCKE	SDRAM 时钟使能	输出	高
SDCS	SDRAM 控制器片选线	输出	低
BA[1:0]	分组选择	输出	
SDWE	SDRAM 写使能	输出	低
RAS - CAS	行列信号	输出	低
NWR[3:0]	写信号	输出	低
NBS[3:0]	字节屏蔽信号	输出	低
SDA10	SDRAM 地址 10 线	输出	



Table 37. I/O 口线说明

名称	功能	类型	工作电平
	Burst Flash 控制器		
BFCK	Burst Flash 时钟	输出	
BFCS	Burst Flash 片选线	输出	低
BFAVD	Burst Flash 地址有效信号	输出	低
BFBAA	Burst Flash 地址超前信号	输出	低
BFOE	Burst Flash 输出使能	输出	低
BFRDY	Burst Flash 就绪信号	输入	高
BFWE	Burst Flash 写使能	输出	低

信号间并不直接复用逻辑而是取决于当前使用的存储控制器。

Table 38 给出三个存储控制器与 EBI 引脚间的连接。

Table 38. EBI 引脚与存储控制器 I/O 线连接

EBI 引脚	SDRAMC I/O 线	BFC I/O 线	SMC I/O 线
NWR1/NBS1/CFIOR	NBS1	不支持	NWR1/NUB
A0/NBS0	不支持	不支持	A0/NLB
A1	不支持	A0	A1
A[11:2]	A[9:0]	A[10:1]	A[11:2]
SDA10	A10	不支持	不支持
A12	不支持	A11	A12
A[14:13]	A[12:11]	A[13:12]	A[14:13]
A[25:15]	不支持	A[24:14]	A[25:15]
D[31:16]	D[31:16]	不支持	不支持
D[15:0]	D[15:0]	D[15:0]	D[15:0]

应用实例

硬件接口

Table 39 给出 EBI 引脚与各存储控制器上的外部器件间的关系。

Table 39. EBI 引脚与外部器件连接

	连接器件引脚						
引脚	8 位静态器件	2 x 8 位静态器件	16 位静态器件	Burst Flash 器件	SDRAM	CompactFlash	SmartMedia 或 NAND Flash
控制器	SMC			BFC	SDRAMC	SMC	
D0 - D7	D0 - D7	D0 - D7	D0 - D7	D0 - D7	D0 - D7	D0 - D7	AD0 - AD7
D8 - D15	–	D8 - D15	D8 - D15	D8 - D15	D8 - D15	D8 - 15	–
D16 - D31	–	–	–	–	D16 - D31	–	–
A0/NBS0	A0	–	NLB	–	DQM0	A0	–

Table 39. EBI 引脚与外部器件连接

引脚	连接器件引脚						
	8 位静态器件	2 x 8 位静态器件	16 位静态器件	Burst Flash 器件	SDRAM	CompactFlash	SmartMedia 或 NAND Flash
控制器	SMC			BFC	SDRAMC	SMC	
A1/NWR2/NBS2	A1	A0	A0	A0	DQM2	A1	–
A2 - A9	A2 - A9	A1 - A8	A1 - A8	A1 - A8	A0 - A7	A2 - A9	–
A10	A10	A9	A9	A9	A8	A10	–
A11	A11	A10	A10	A10	A9	–	–
SDA10	–	–	–	–	A10	–	–
A12	A12	A11	A11	A11	–	–	–
A13 - A15	A13 - A15	A12 - A14	A12 - A14	A12 - A14	A11 - A13	–	–
A16/BA0	A16	A15	A15	A15	BA0	–	–
A17/BA1	A17	A16	A16	A16	BA1	–	–
A18 - A20	A18 - A20	A17 - A19	A17 - A19	A17 - A19	–	–	–
A21	A21	A20	A20	A20	–	–	CLE(4)
A22	A22	A21	A21	A21	–	REG(3)	ALE(4)
A23 - A24	A23 - A24	A22 - A23	A22 - A23	A22 - A23	–	–	–
A25	A25	A24	A24	A24	–	CFRNW(1)	–
NCS0/BFCS	CS	CS	CS	CS	–	–	–
NCS1/SDCS	CS	CS	CS	–	CS	–	–
NCS2	CS	CS	CS	–	–	–	–
NCS3/SMCS	CS	CS	C S	–	–	–	–
NCS4/CFCS	CS	CS	CS	–	–	CFCS(1)	–
NCS5/CFCE1	CS	CS	CS	–	–	CE1	–
NCS6/CFCE2	CS	CS	CS	–	–	CE2	–
NRD/NOE/CFOE	OE	OE	OE	–	–	OE	
NWR0/NWE/CFWE	WE	WE(5)	WE	–	–	WE	
NWR1/NBS1/CFIOR	WE	WE(5)	NUB	–	DQM1	IOR	–
NWR3/NBS3/CFIOW	–	–	–	–	DQM3	IOW	–
BFCK	–	–	–	CK	–	–	–
BFAVD	–	–	–	AVD	–	–	–
BFBAA/SMWE	–	–	–	BAA	–	–	WE
BFOE	–	–	–	OE	–	–	–
BFRDY/SMOE	–	–	–	RDY	–	–	OE
BFWE	–	–	–	WE	–	–	–
SDCK	–	–	–	–	CLK	–	–
SDCKE	–	–	–	–	CKE	–	–

Table 39. EBI 引脚与外部器件连接

引脚	连接器件引脚						
	8 位静态器件	**2 x 8 位静态器件**	**16 位静态器件**	**Burst Flash 器件**	**SDRAM**	**CompactFlash**	**SmartMedia 或 NAND Flash**
控制器	**SMC**			**BFC**	**SDRAMC**	**SMC**	
RAS	–	–	–	–	RAS	–	–
CAS	–	–	–	–	CAS	–	–
SDWE	–	–	–	–	WE	–	–
NWAIT	–	–	–	–	–	WAIT	–
Pxx(2)	–	–	–	–	–	CD1 or CD2	–
Pxx(2)	–	–	–	–	–	–	CE
Pxx(2)	–	–	–	–	–	–	RDY

Notes:
1. 不与 CompactFlash 插槽直接连接。允许 EBI 数据总线与 CompactFlash 插槽间的双向缓冲。
2. 任意 PIO 线。
3. CompactFlash 中的 REG 信号可由以下地址位进行驱动：A24、 A22 ~ A11，详见 "CompactFlash 支持" on page 140。
4. SmartMedia 器件的 CLE 与 ALE 信号可由任意地址位进行驱动，详见 "SmartMedia 与 NAND Flash 支持" on page 143。
5. NWR1 使能高字节写； NWR0 使能低字节写。

连接实例

Figure 32 给出 EBI 与外部器件的连接图。

Figure 32. EBI 与存储器件连接

EBI
D0-D31
RAS
CAS
SDCK
SDCKE
SDWE
A0/NBS0
NWR1/NBS1
A1/NWR2/NBS2
NWR3/NBS3
NRD/NOE
NWR0/NWE
SDA10
A2-A15
A16/BA0
A17/BA1
A18-A25
NCS0/BFCS
NCS1/SDCS
NCS2
NCS3
NCS4
NCS5
NCS6
NCS7
BFCLK
BFOE
BFWE
BFAVD
BFRDY

2M x 8 SDRAM
D0-D7 — D0-D7
CS
CLK
CKE
SDWE — WE
RAS
CAS
NBS0 — DQM
A0-A9, A11 — A2-A11, A13
A10 — SDA10
BA0 — A16/BA0
BA1 — A17/BA1

2M x 8 SDRAM
D8-D15 — D0-D7
CS
CLK
CKE
SDWE — WE
RAS
CAS
NBS1 — DQM
A0-A9, A11 — A2-A11, A13
A10 — SDA10
BA0 — A16/BA0
BA1 — A17/BA1

2M x 8 SDRAM
D16-D23 — D0-D7
CS
CLK
CKE
SDWE — WE
RAS
CAS
NBS2 — DQM
A0-A9, A11 — A2-A11, A13
A10 — SDA10
BA0 — A16/BA0
BA1 — A17/BA1

2M x 8 SDRAM
D24-D31 — D0-D7
CS
CLK
CKE
SDWE — WE
RAS
CAS
NBS3 — DQM
A0-A9, A11 — A2-A11, A13
A10 — SDA10
BA0 — A16/BA0
BA1 — A17/BA1

2M x 16 Burst Flash
D0-D15 — D0-D15
CE
CLK
OE
WE
AVD
RDY
A0-A20 — A1-A21

128K x 8 SRAM
D0-D7 — D0-D7
A0-A16 — A1-A17
CS
NRD/NOE — OE
A0/NWR0/NBS0 — WE

128K x 8 SRAM
D8-D15 — D0-D7
A0-A16 — A1-A17
CS
NRD/NOE — OE
NWR1/NBS1 — WE



相关产品

I/O 线

外部总线接口使用的引脚可能与 PIO 线复用。程序员必须先对 PIO 控制器编程来分配外部总线接口引脚的外设功能。若应用程序没有使用外部总线接口的 I/O 口线，它们可被 PIO 控制器用作其它用途。

功能说明

EBI 在内部 ASB 总线 (由存储控制器处理) 与外部存储器或外设间传输数据。它控制外部地址、数据及控制总线的波形与参数，它由以下部分组成：

- 静态存储控制器 (SMC)
- SDRAM 控制器 (SDRAMC)
- Burst Flash 控制器 (BFC)
- 芯片选择分配特性，给外部器件分配一个 ASB 地址空间
- 一个多路控制器电路，不同存储控制器间共享引脚
- 可编程 CompactFlash 支持逻辑
- 可编程 SmartMedia 与 NAND Flash 支持逻辑

总线复用

EBI 通过存储域操作函数需求的复用逻辑提供的一套控制信号来共享 32 位数据线 ,26 位地址线及控制信号。

没有外部访问时，复用管理需特别小心，以保证地址与输出控制线处在一个稳定的状态。在定义存储控制器数据流动时间时应考虑复用器的设计。此外， SDRAM 的刷新周期由 SDRAM 控制器独立执行，不会延迟其它外部存储控制器的访问。还有就是它避免了高峰访问 burst Flash 相同页面时发生中断，从而避免了重启高延时的访问需要。

上拉控制

EBI 允许使能不与 PIO 控制器线复用的数据总线上的片上上拉电阻。上拉电阻在复位后使能。设置 DBPUC 位将禁用 D0 ~ D15 线的上拉电阻。通过对适当的 PIO 控制器编程可使能 D16 ~ D31 的上拉电阻。

静态存储控制器

详见 SMC “Overview” on page 151。

SDRAM 控制器

详见 “ 概述 ” on page 133 中 SDRAMC 的说明。

Burst Flash 控制器

详见 “Overview” on page 209 BFC。

CompactFlash 支持

EBI 集成了 CompactFlash 器件接口电路。

CompactFlash 逻辑由 NCS4 地址空间的静态存储控制器 (SMC) 驱动。将片选任务寄存器的 CS4A(见See “EBI片选任务寄存器” on page 147.)编程为合适值可使能该逻辑。通过访问保留给 NCS4的地址空间可访问外部CompactFlash器件(即 ,0x5000 0000与0x5FFF FFFF间地址)。

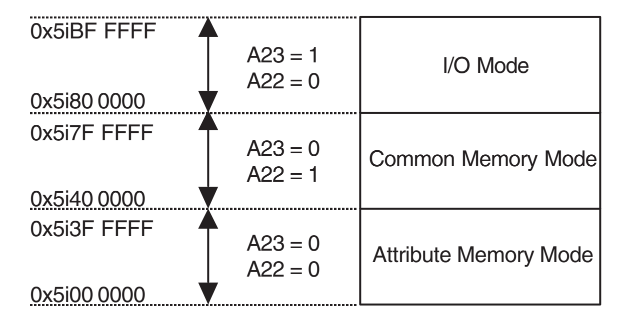
当复用 CFCE1 与 CFCE2 信号时， NCS5 与 NCS6 信号无效。执行对保留给 NCS5 与 NCS6 的地址空间的访问 (即， 0x6000 0000 与 0x7FFF FFFF 间地址)，其结果无法预料。

不支持 IDE 模式，在 I/O 模式下，信号 _IOIS16 没有使用。

I/O 模式，通用存储器模式与标志存储模式

在 NCS4 地址空间中 , 当前的传输地址用来辨别 I/O 模式 , 通用存储模式还是标志存储模式。传输地址的 A23 位用作 I/O 模式选择。CompactFlash 器件不需要任何 EBI 地址位 (即位 A24 或位 A22 ~ A11) 就能分离通用存储模式和标志存储模式 。例如使用 A22 位，地址映射如 Figure 33 所示，图中 “i” 表示任意的十六进制数。

Figure 33. 地址映射示例



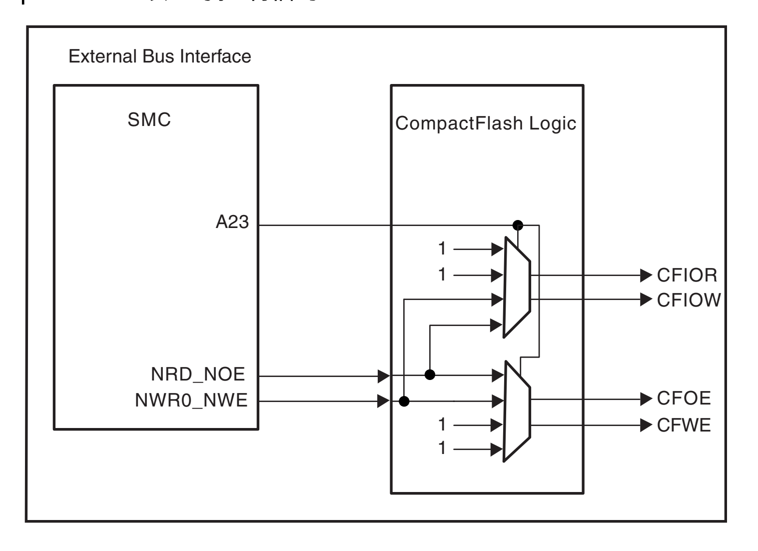
Note: 上例中， EBI 的 A22 引脚可用来驱动 CompactFlash 器件的 REG 信号。

读 / 写信号

I/O 模式下 ,CompactFlash 逻辑驱动 CFIOR 与 CFIOW 信号上 SMC 的读写信号 , 此时 CFOE 与 CFWE 信号失效。同样在通用存储模式和标志存储模式下驱动 CFOE 与 CFWE 信号上的 SMC，CFIOR 与 CFIOW 信号失效。 Figure 34 on page 141 给出该逻辑的图示。

通过设置 NCS4 片选地址设置与保持时间为合适值 , 可支持标志存储模式 , 通用存储模式及 I/O 模式 。关于这些信号的详细波形，请参阅静态存储控制器的 “Setup and Hold Cycles” on page 164 部分。

Figure 34. CompactFlash 读 / 写控制信号



访问类型

CFCE1与CFCE2信号使能CompactFlash器件数据总线由上或是由下访问 ,见Table 40。只有当 NCS4 引脚上的 SMC 配置为驱动 8 位存储器时才可进行奇字节字节访问。NCS4 地址空间中的片选寄存器 (“SMC Chip Select Registers” on page 186 中的 DBW 域) 必须如 Table 40 所示进行

设置以使能所需访问类型。CFCE1 与 CFCE2 波形对于 NCS4 波形都是相同的，详见静态存储控制器 “Overview” on page 151。

Table 40. 自上与自下字节访问

访问	CFCE2	CFCE1	A0	D[15:8]	D[7:0]	SMC_CSR4 (DBW)
字节读 / 写访问	1	0	0	不在意 / 高 Z	偶字节	8 位或 16 位
	1	0	1	不在意 / 高 Z	奇字节	8 位
奇字节读 / 写访问	0	1	X	奇字节	不在意 / 高 Z	16 位
半字读 / 写访问	0	0	X	奇字节	偶字节	16 位

EBI 引脚上 CompactFlash 信号的复用

Table 41与Table 42 on page 142 阐明了CompactFlash逻辑信号与EBI引脚上的其它EBI信号的复用。Table 41 中的 EBI 引脚在片选任务寄存器置位后 (见 See “EBI 片选任务寄存器 ” on page 147.) 只与 CompactFlash 接口连接。这些引脚不允许由其它存储器件驱动。

Table 42 on page 142 中的 EBI 引脚在 CompactFlash 接口使能时 (CS4A = 1) 与所有存储域保持共享。

Table 41. 专门的 CompactFlash 接口复用

	CS4A = 1	CS4A = 0
引脚	CompactFlash 信号	EBI 信号
NCS4/CFCS	CFCS	NCS4
NCS5/CFCE1	CFCE1	NCS5
NCS6/CFCE2	CFCE2	NCS6

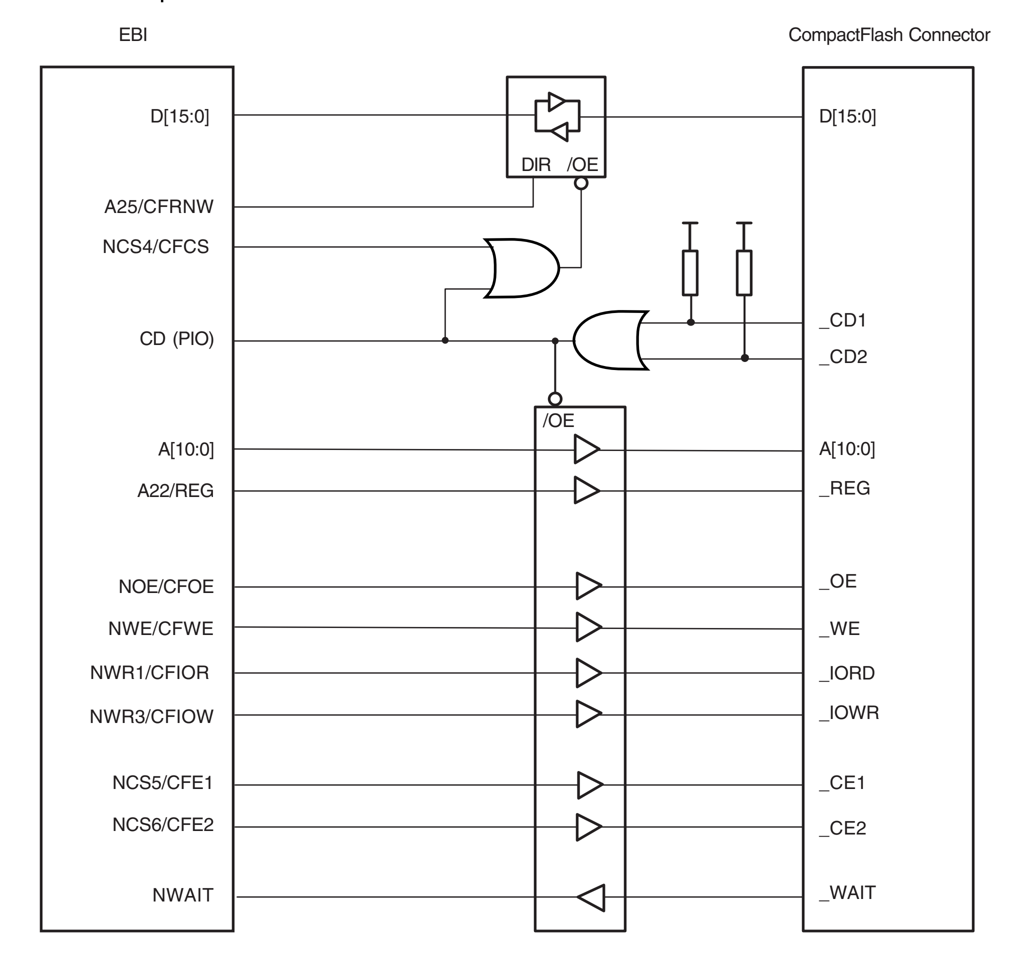
Table 42. 共享的 CompactFlash 接口复用

	对 CompactFlash 器件的访问	对其它 EBI 器件的访问
引脚	CompactFlash 信号	EBI 信号
NOE/NRD/CFOE	CFOE	NRD/NOE
NWR0/NWE/CFWE	CFWE	NWR0/NWE
NWR1/NBS1/CFIOR	CFIOR	NWR1/NBS1
NWR3/NBS3/CFIOW	CFIOW	NWR3/NBS3
A25/CFRNW	CFRNW	A25

CompactFlash 应用实例

Figure 35 给出 CompactFlash 应用实例。CFCS 与 CFRNW 信号不直接与 CompactFlash 插槽连接，但控制 EBI 与 CompactFlash 器件间缓冲的方向及输出使能。 CFCS 定时信号与 NCS4 信号相同。此外，传输过程中 CFRNW 信号与地址总线保持有效 。CompactFlash _WAIT 信号连接在静态存储控制器的 NWAIT 输入上，更多的关于波形与定时的内容见静态存储控制器 “ 概述 ” on page 133。

Figure 35. CompactFlash 应用实例



SmartMedia 与 NAND Flash 支持

EBI 集成了 SmartMedia 与 NAND Flash 器件接口电路。

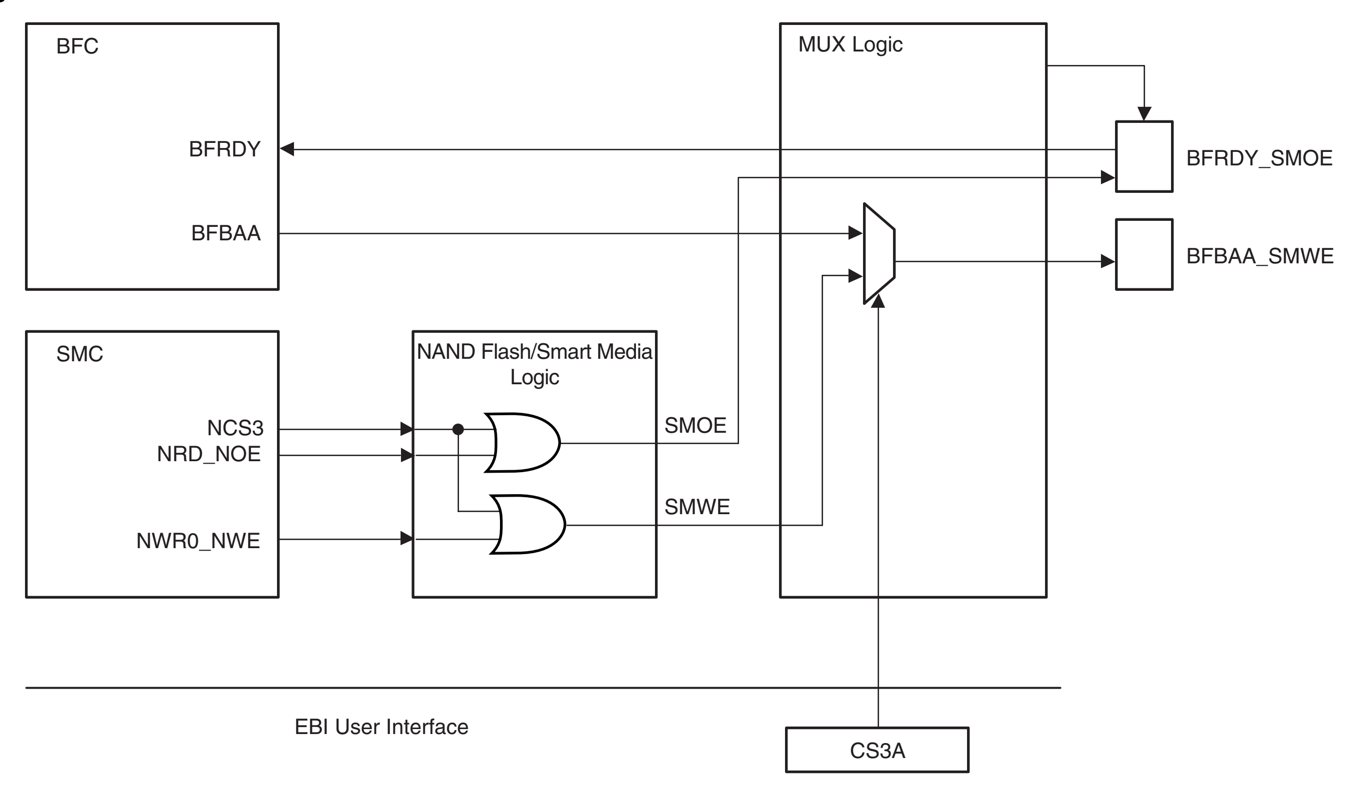
静态存储控制器的 NCS3 地址空间逻辑驱动 SmartMedia。对片选任务寄存器的 CS3A 编程到合适值使能 SmartMedia 逻辑 (见 See “EBI 片选任务寄存器 ” on page 147.)。通过访问保留给 NCS3 的地址空间来访问外部 SmartMedia 器件 (即 ,0x4000 0000 与 0x4FFF FFFF 间地址)。

当 NCS3 信号激活时， SmartMedia 逻辑驱动 SMOE 与 SMWE 信号上的 SMC 的读写命令信号。一旦向 NCS3 地址空间传输失败， SMOE 与 SMWE 失效 , 更多细节见静态存储控制器 “Overview” on page 151。

SMWE 与 SMOE 信号与 Burst Flash 控制器的 BFRDY 与 BFBAA 信号复用。该复用由片选任务寄存器 CS3A 域的 EBI 多路复用逻辑控制 (See “EBI 片选任务寄存器 ” on page 147.)。该逻辑还控制 BFRDY/SMOE 焊盘方向。

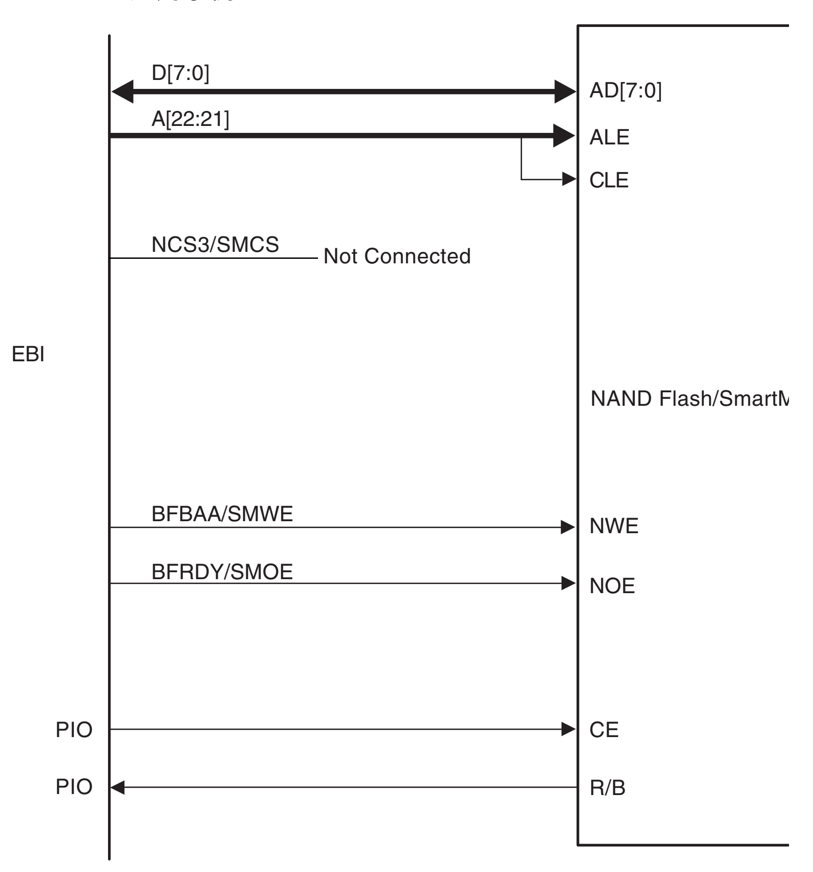


Figure 36. SmartMedia 信号在 EBI 引脚上的复用



SmartMedia器件的地址锁存使能与命令锁存使能信号由EBI地址总线的A22与A21位驱动。用户应注意 EBI 地址总线的任意位均可作此目的。 SmartMedia 器件数据总线上的命令、地址或数据字通过使用它们在 NCS3 地址空间中的地址进行识别。芯片使能 (CE) 信号及就绪 / 忙 (R/B) 信号与 PIO 线连接。当 NCS3 未选定时 ,CE 信号保持 ,以防器件返回等待模式。当 SmartMedia 器件激活时，由于 SMOE 与 SMWE 信号分别与 BFRDY 与 BFBAA 信号复用，对于支持 burst Flash 器件将出现一些功能限制。

Figure 37. SmartMedia 应用示例





外部总线接口 (EBI) 用户接口

AT91RM9200 EBI 用户接口基地址 : 0xFFFF FF60

Table 43. 外部总线接口存储器映射

偏移	寄存器	名称	访问	复位状态
0x00	片选任务寄存器	EBI_CSA	读 / 写	0x0
0x04	配置寄存器	EBI_CFGR	读 / 写	0x0
0x08	保留	–		
0x0C	保留	–		
0x10 - 0x2C	SMC 用户接口	见 “Static Memory Controller (SMC) User Interface” on page 185		
0x30 - 0x5C	SDRAMC 用户接口	见 See “SDRAM Controller (SDRAMC) User Interface” on page 201.		
0x60	BFC 用户接口	见 See “Burst Flash Controller (BFC) User Interface” on page 221.		
0x64 - 0x9C	保留			

EBI 片选任务寄存器

寄存器名称 : EBI_CSA
访问类型 : 读 / 写
复位值 : 0x0
偏移 : 0x0
绝对地址 : 0xFFFF FF60

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	CS4A	CS3A	–	CS1A	CS0A

- **CS0A: 片选 0 配置**

0 = 片选 0 配置为静态存储控制器

1 = 片选 0 配置为 Burst Flash 控制器

- **CS1A: 片选 1 配置**

0 = 片选 1 配置为静态存储控制器

1 = 片选 1 配置为 SDRAM 控制器

- **CS3A: 片选 3 配置**

0 = 片选 3 只配置为静态存储控制器且 NCS3 行为由 SMC 定义

1 = 片选 3 配置为静态存储控制器且激活 SmartMedia 逻辑

- **CS4A: 片选 4 配置**

0 = 片选 4 配置为静态存储控制器且 NCS4、 NCS5 及 NCS6 行为由 SMC 定义

1 = 片选 4 配置为静态存储控制器且激活 CompactFlash 逻辑

对保留给 NCS5 及 NCS6 的地址空间进行访问，其结果无法预料。



EBI 配置寄存器

寄存器名称 : EBI_CFGR
访问类型 : 读 / 写
复位值 : 0x0
偏移 : 0x04
绝对地址 : 0xFFFF FF64

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–		DBPUC

- **DBPUC: 数据总线上拉配置**

0 = [D15:0] 数据总线内部上拉到 VDDIOM 电压

1 = [D15:0] 数据总线没有内部上拉

静态存储控制器 (SMC)

概述

静态存储控制器 (SMC) 产生信号来控制外部静态存储器或外设的访问。 SMC 可编程地址达 512M字节。它有8个片选及一个26位地址总线。16位数据总线能配置与8位或16位外部器件连接。独立的读写控制信号允许存储器与外设直接连接。SMC 支持不同的允许单时钟周期存储器访问的访问协议。它还提供外部等待请求能力。

SMC 主要特性为 :

- 外部存储器映射 , 512-M 字节地址空间
- 8 个片选线
- 8 位或 16 位数据总线
- 引导存储器重映射
- 支持多路访问模式
  - 字节写或字节选择线
  - 每个存储器组有两个不同的读协议
- 多器件适应性
  - LCD 模块适用
  - 可编程启动时间读 / 写
  - 可编程保持时间读 / 写
- 多等待状态管理
  - 可编程等待状态产生
  - 外部等待请求
  - 可编程数据流动时间

方框图

Figure 38. 静态存储控制器方框图

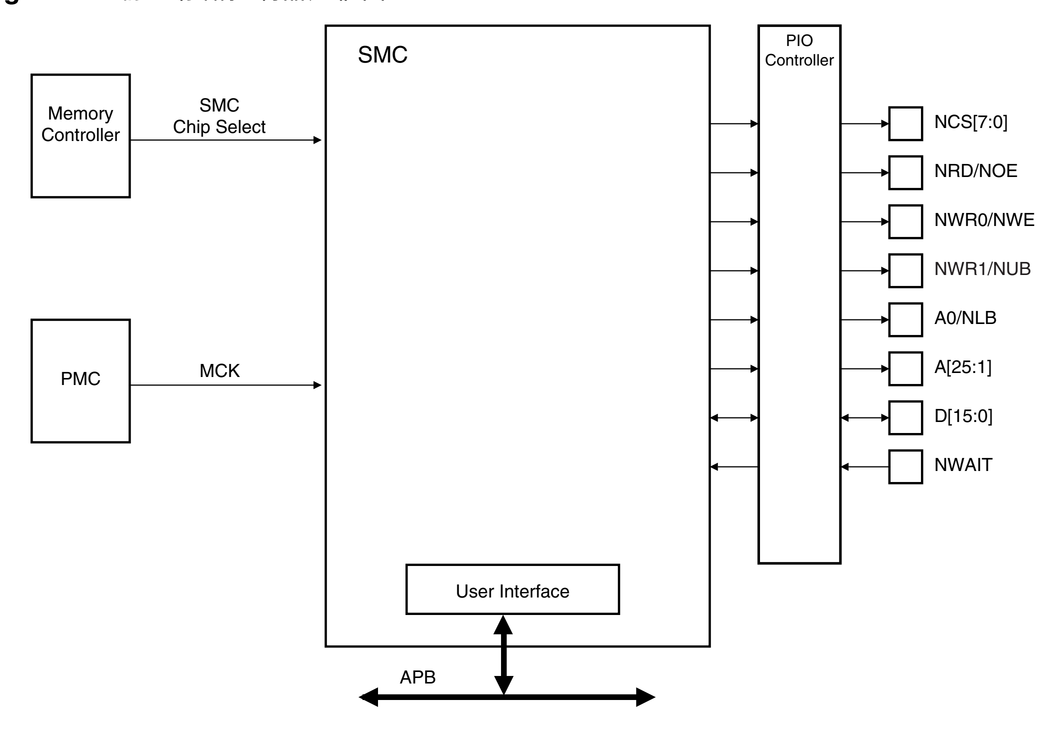


Table 44. I/O 线说明

名称	说明	类型	工作电平
NCS[7:0]	静态存储控制器片选线	输出	低
NRD/NOE	读 / 输出使能信号	输出	低
NWR0/NWE	写 0/ 写使能信号	输出	低
NWR1/NUB	写 1/ 高字节选择信号	输出	低
A0/NLB	地址位 0/ 低字节选择信号	输出	低
A[25:1]	地址总线	输出	
D[15:0]	数据总线	I/O	
NWAIT	外部等待信号	输入	低

复用信号及其功能见 Table 45。

Table 45. 静态存储控制器复用信号

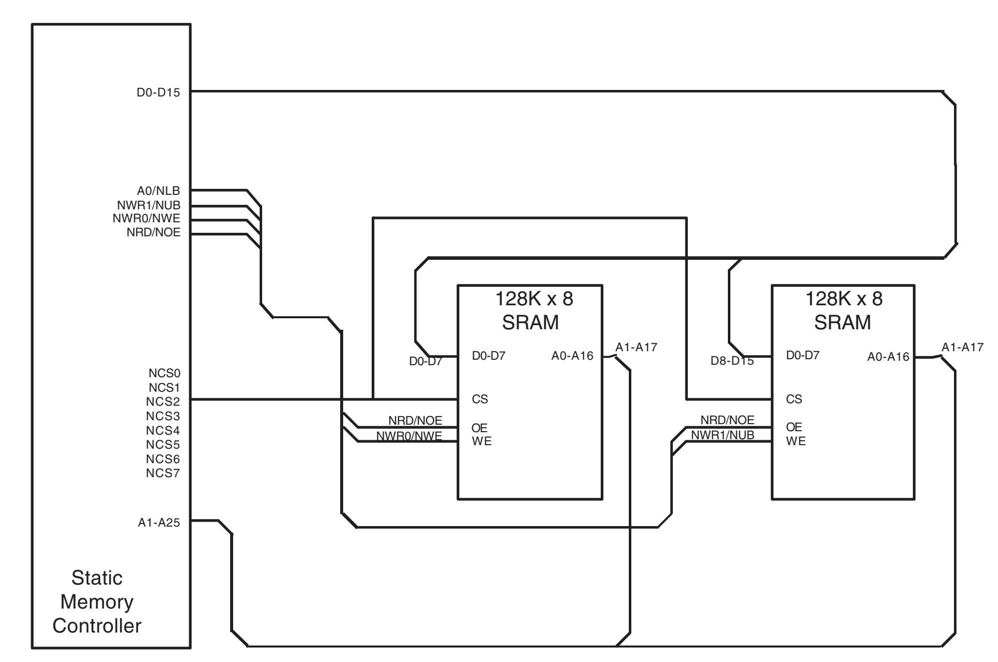
复用信号		相关功能
A0	NLB	8 位或 16 位数据总线，见 " 数据总线宽度 " on page 152。
NRD	NOE	字节写或字节选择访问，见 " 写访问类型 " on page 154。
NWR0	NWE	字节写或字节选择访问，见 " 写访问类型 " on page 154。
NWR1	NUB	字节写或字节选择访问，见 " 写访问类型 " on page 154。

应用实例

硬件接口

Figure 39 给出静态存储器件与 SMC 连接实例。

Figure 39. SMC 与静态存储器件连接图



相关产品

I/O 线

连接静态存储控制器的引脚也可复用为 PIO 线。必须首先对 PIO 控制器编程，分配静态存储控制器引脚给其外设功能。若程序中未使用静态存储控制器的 I/O 线，则它们被 PIO 控制器用于其它用途。



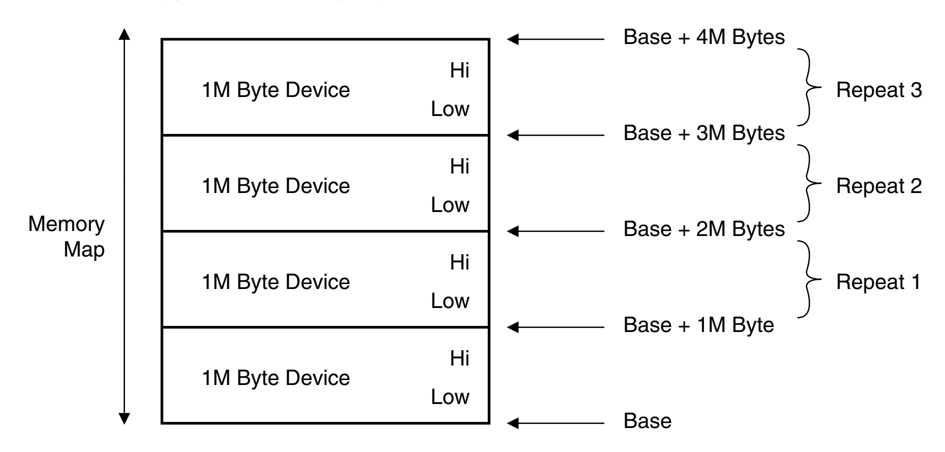
功能说明

外部存储器接口

外部存储器映射

存储器映射由硬件定义，将内部 32 位地址空间与外部 26 位地址总线联系起来。注意 A[25:0] 主要用于访问 8 位存储器；而 A[25:1] 用于访问 16 位存储器。若物理存储器空间小于页空间，则在页内循环重复调度。 SMC 处理页内存储器件的有效访问，见 Figure 40。

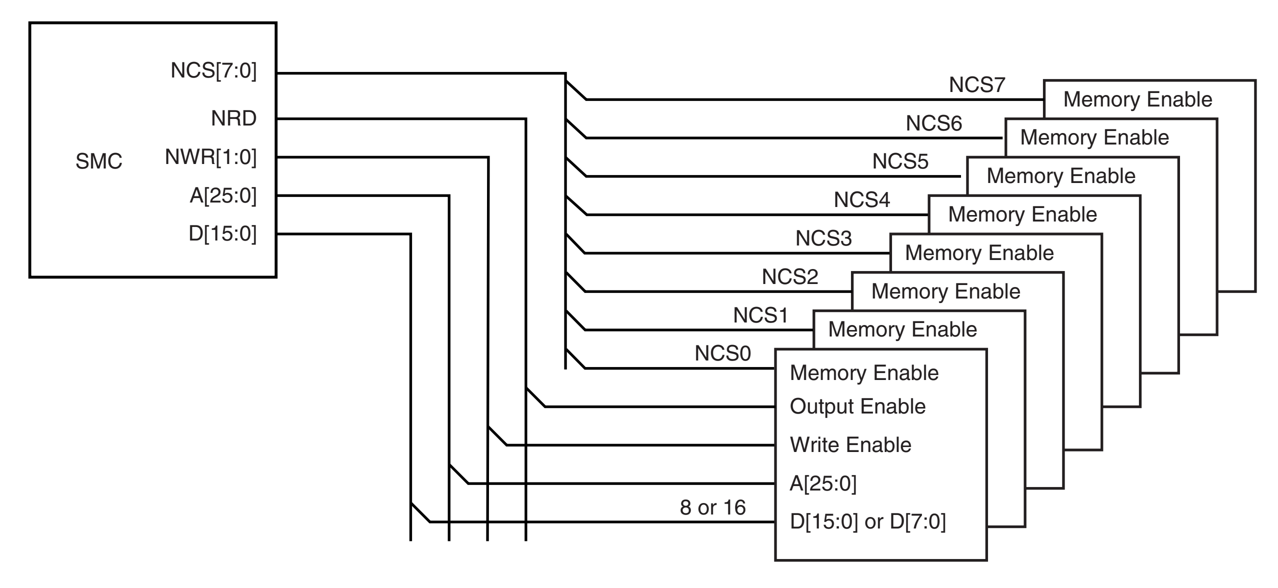
Figure 40. 一旦外部存储器空间小于页空间



片选线

静态存储控制器提供 8 个片选线：NCS0 ～ NCS7。

Figure 41. 存储器与 8 个外部器件连接 (1)



Note: 1. 每个器件的最大地址空间为 512M。

数据总线宽度

每个片选的数据总线宽度为 8 位或 16 位。SMC_CSR 中的 DBW 控制相应的片选，见 "SMC 片选寄存器 " on page 184。

Figure 42 给出 NCS2 上如何连接一个 512K x 8 位存储器 (DBW = 10)。

Figure 42. 存储器与一个 8 位数据器件的连接

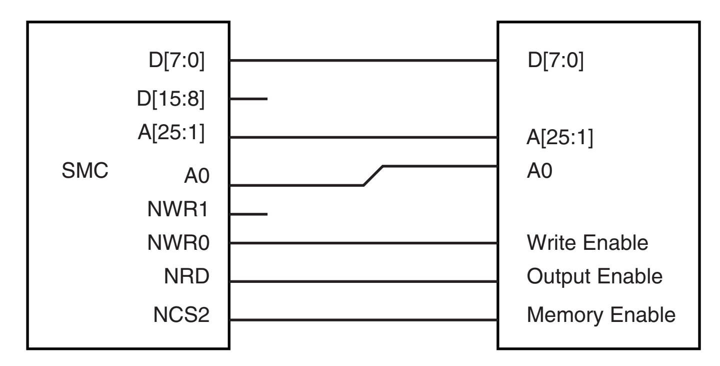
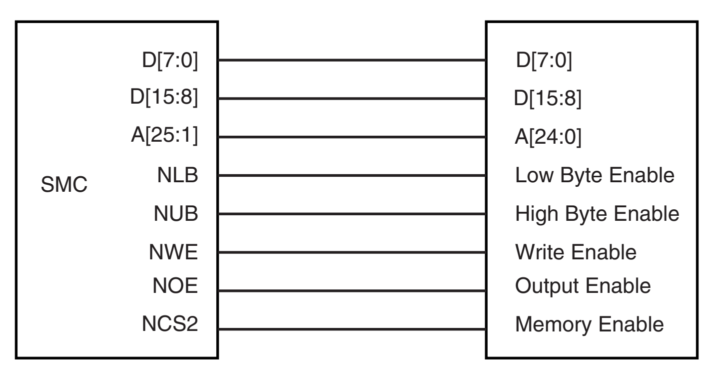


Figure 43 给出 NCS2 上如何连接一个 512K x 16 位存储器 (DBW = 01)。

Figure 43. 存储器与一个 16 位数据器件的连接





写访问

写访问类型

16 位数据总线片选写访问有两种类型：

- 字节写访问支持两字节写单个读信号。
- 字节选择访问提供两字节选择线选择高位与 / 或低位字节，且区分读与写信号。

该项由 SMC_CSR 中相应的片选域 BAT 控制，见 “SMC 片选寄存器 ” on page 184。

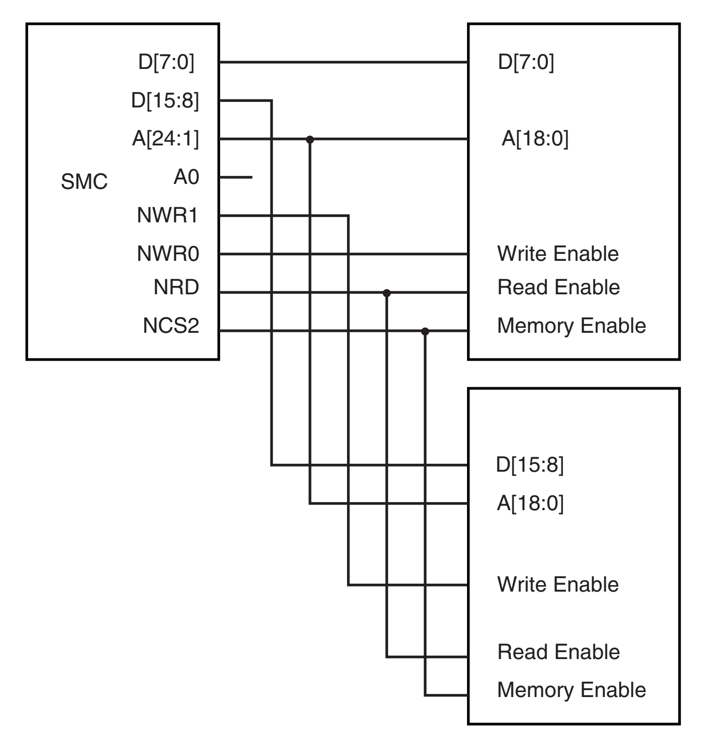
字节写访问

字节写访问将 2 x 8 位器件作为 16 位存储页。

- 不使用信号 A0/NLB。
- 使用信号 NWR1/NUB 用作 NWR1 并使能高字节写入。
- 使用信号 NWR0/NWE 用作 NWR0 并使能低字节写入。
- 使用信号 NRD/NOE 用作 NRD 并使能半字与字节读取。

Figure 44 给出如何在 NCS2 上并行连接两个 512K x 8 位器件 (BAT = 0)。

Figure 44. 存储器与 2 x 8 位数据器件的连接



字节选择访问

字节选择访问用于连接存储页中的 16 位器件。

- 信号 A0/NLB 用作 NLB 且使能低字节读与取操作。
- 信号 NWR1/NUB 用作 NUB 且使能高字节读与取操作。
- 信号 NWR0/NWE 用作 NWE 且使能字节或半字写。
- 信号 NRD/NOE 用作 NOE 且使能字节或半字读。

Figure 45 给出字节或半字访问 (例如，SRAM 型器件) NCS2 时 16 位器件连接图 (BAT = 1)。

Figure 45. 字节或半字访问的 16 位数据器件连接图

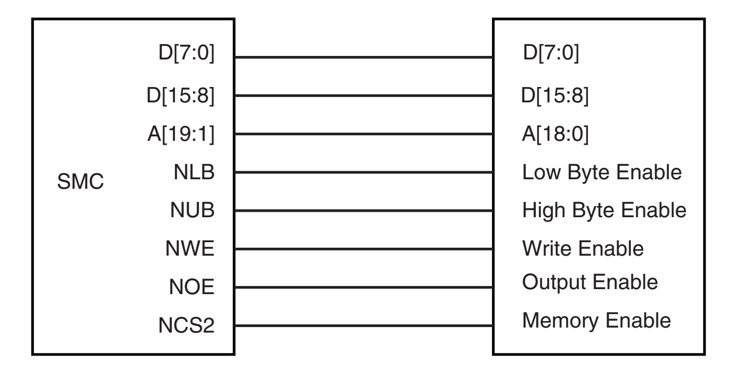
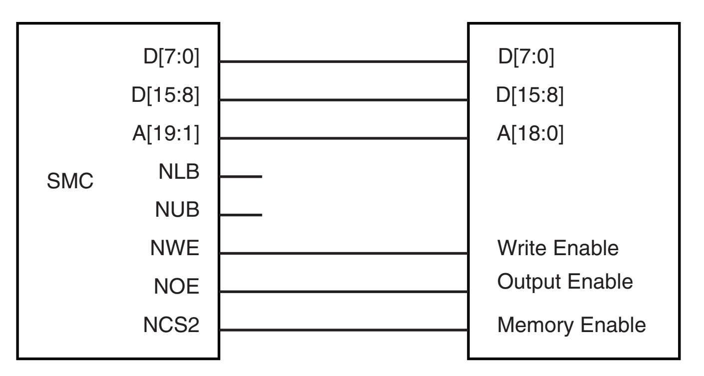


Figure 46 给出没有字节访问 (例如，Flash 器件类型)NCS2 时的 16 位器件连接 (BAT = 1)。

Figure 46. 没有字节访问的 16 位器件连接图



写数据保持时间

在写周期中，在 MCK 上升沿后数据输出有效且保持到 NWE 下降沿。写访问时，在 NWE 下降沿后，数据在总线上保持 1/2 MCK 周期，见 Figure 47 与 Figure 48。

Figure 47. 0 等待状态写访问

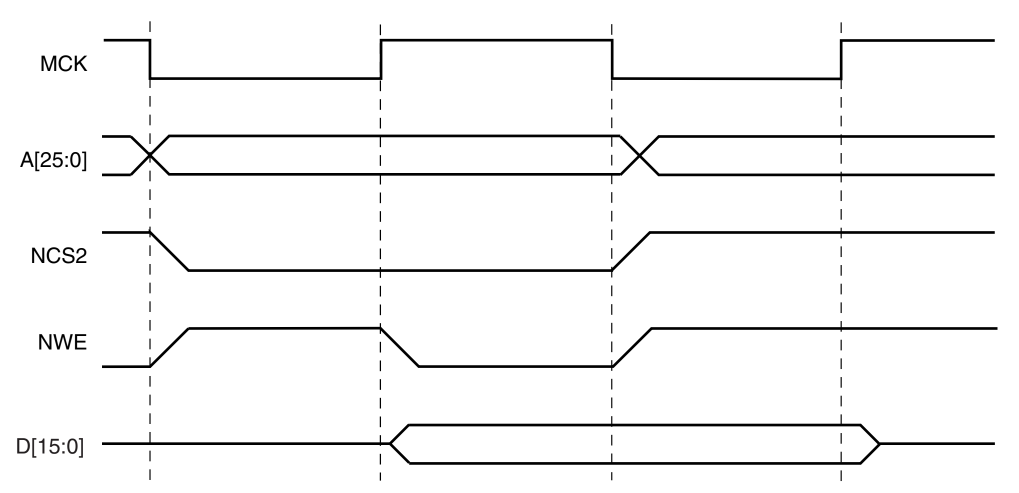
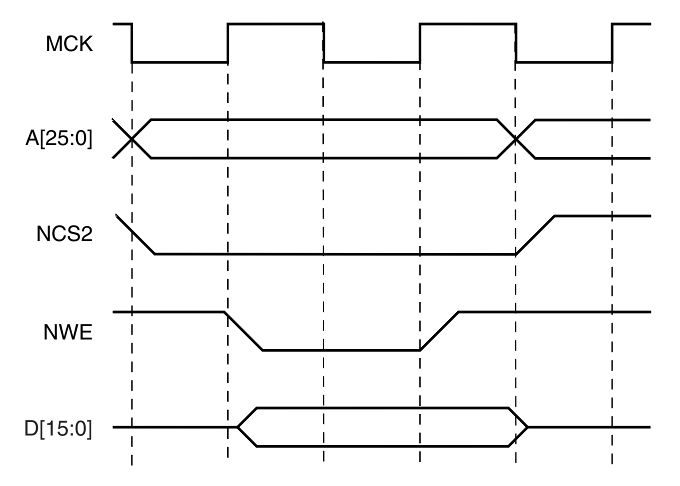


Figure 48. 1 等待状态写访问



读访问

读协议

SMC 提供两种对外部存储器读访问协议 : 标准与预读。两种协议的不同在于 NRD 信号的表现。

读访问时，对两种协议而言 NWE 行为相同。在后半主时钟周期中， NWE 变低 (见 Figure 56 on page 162)。

协议由 SMC_CSR 中的 DRP 域选定 (见 See “SMC 片选寄存器 ” on page 184.)。复位后的默认读协议为标准读协议。

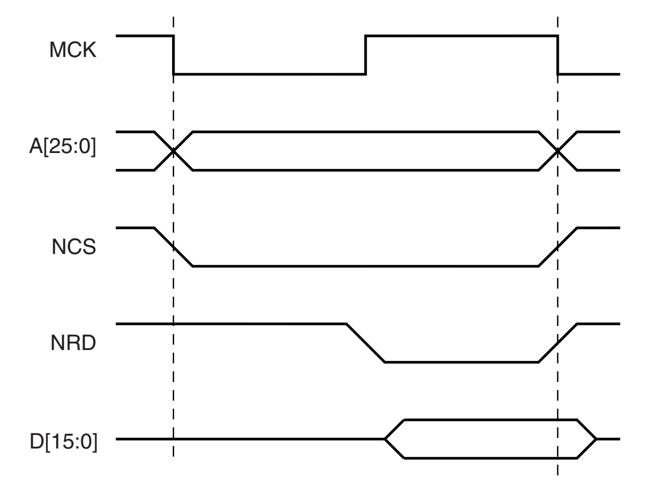
Note: 由于 NRD 与 NOE 信号波形相同 , 因此在下面的说明中 , 只讨论 NRD。同样 , 用 NWE 代替 NWE、NWR0 及 NWR1 , 除非 NWR0 及 NWR1 已由其它信号代替。此外 , NCS 代表 NCS[7:0] (见 “I/O 线” on page 151、 Table 44 及 Table 45)。

标准读协议

标准读协议在 NRD 与 NWE 相似时执行一个读循环。 NRD 与 NWE 都是在后半个时钟周期中激活。前半个时钟周期用来确保有足够的时间完成读循环开始前的地址输出及 NCS 访问。

在标准读协议中，外部存储器开始访问时， NCS 置低地址线有效，而 NRD 只会在主机时钟后半个周期中变低以避免总线冲突，见 Figure 49。

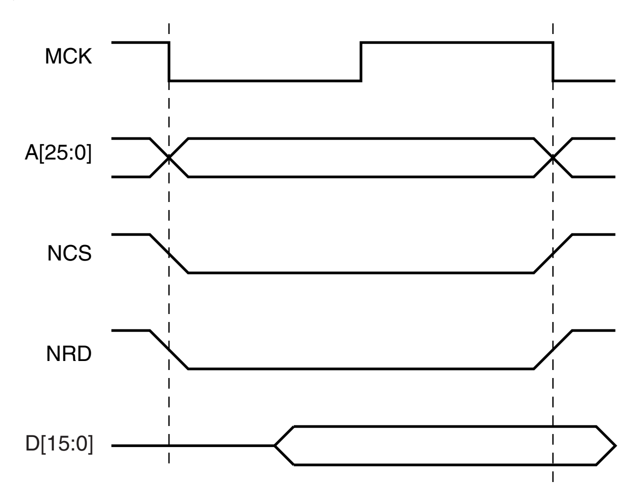
Figure 49. 标准读协议



预读协议

预读协议通过在时钟周期开始时插入 NRD 信号为对存储器的读访问提供了更多的时间。在对相同的存储器成功完成了读访问周期后，NRD 继续保持有效。由于读周期通常会限制外部存储器系统速度，预读协议允许使用一个更快的时钟频率。但为避免出现外部总线竞争，有时需要一个额外的等待状态。

Figure 50. 预读协议



等待状态管理

SMC 可自动插入等待状态。下面列出不同类型的等待状态：

- 标准等待状态
- 额外等待状态
- 数据流动等待状态
- 片选改变等待状态
- 预读等待状态

标准等待状态

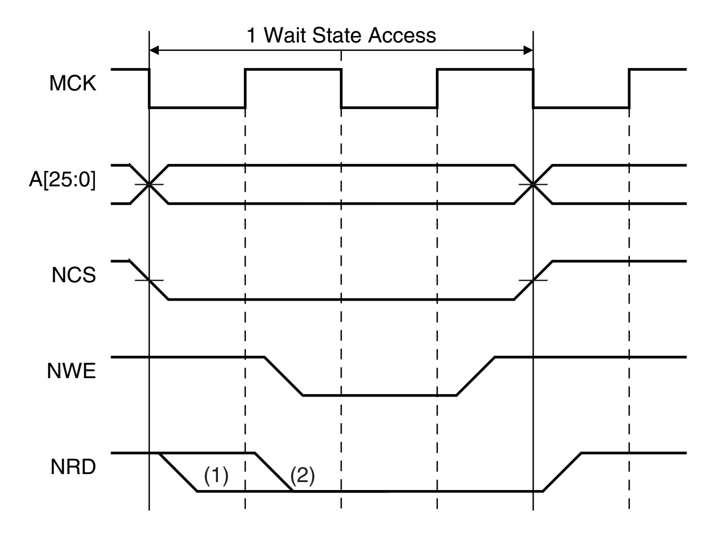
在访问相应的存储器空间时，可对片选编程插入一个或几个等待状态，这可通过在相应的 SMC_CSR中设置WSEN域来实现(见See “SMC片选寄存器” on page 184.)。插入的周期数目由同一寄存器中的 NWS 域确定。

下面是相应的标准等待状态数与 NWE 脉冲为低时时钟周期数：

0 等待状态	1/2 时钟周期
1 等待状态	1 时钟周期

每附加一个等待状态，将增加一个额外的周期。

Figure 51. 一个标准等待状态访问



Notes: 1. 预读协议
2. 标准读协议

外部等待状态

NWAIT 输入引脚用来在超出最大可编程标准等待状态时插入等待状态。若 NWAIT 为低，SMC 增加一个等待状态且输出信号、内部计数器或状态没有改变。当 NWAIT 无效， SMC 完成访问序列。

NWAIT 信号为异步输入。为避免不稳定的问题，必须在使用前将 NWAIT 同步。该操作引起两个周期的延时。

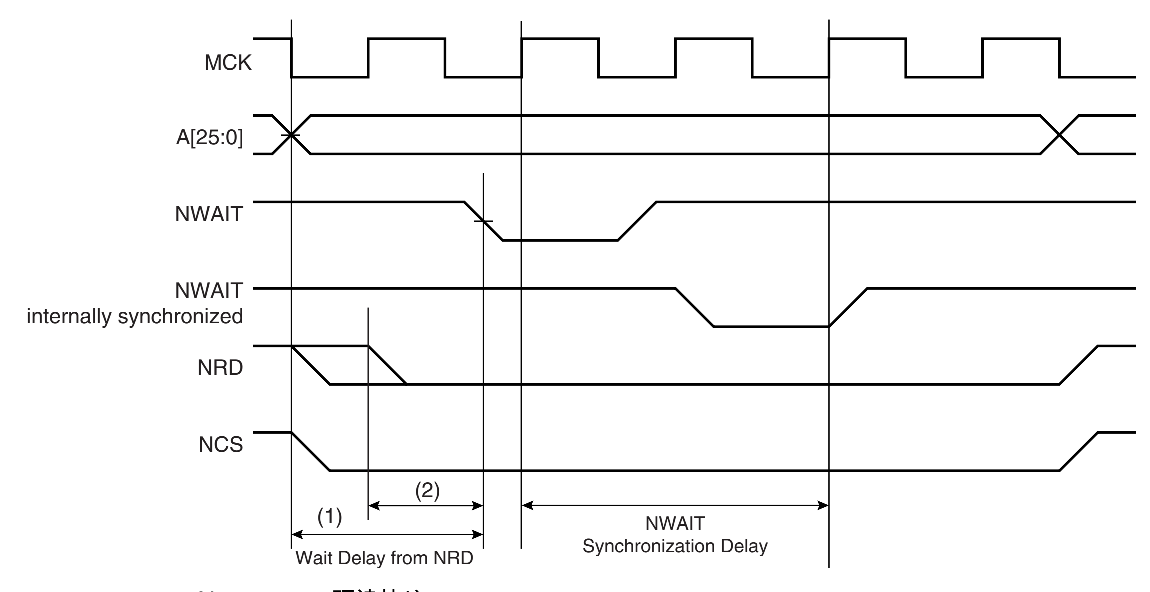
NWS须编程使NWAIT下降与控制信号下降(NRD/NWE)同步计时与延迟，否则SMC无法正常工作。

NWS > Wait Delay from nrd/nwe + external_nwait Synchronization Delay + 1

若在启动或保持时插入 NWAIT， SMC 无法正常工作。

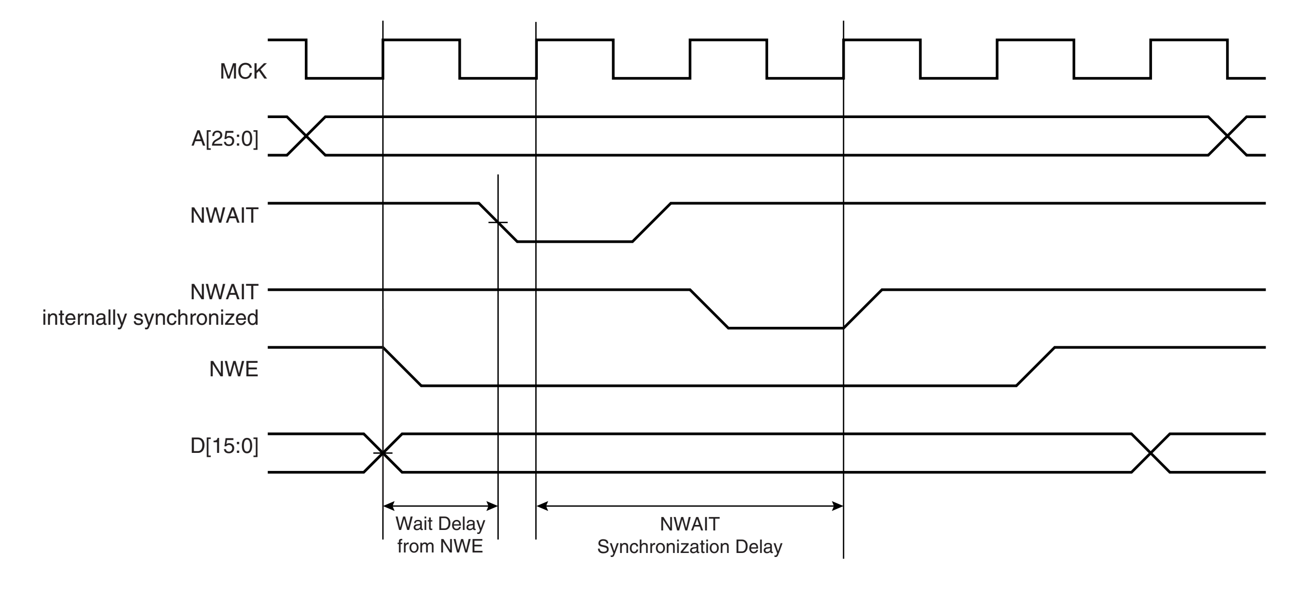


Figure 52. 读访问时 NWAIT 状况



Notes: 1. 预读协议
2. 标准读协议

Figure 53. 写访问时 NWAIT 状况



数据流动等待状态

某些存储器会慢速双方外部总线。对于这些器件，在读访问后并开始对其它外部存储器进行读访问或写操作时需要加入等待状态 (数据流动等待状态)。

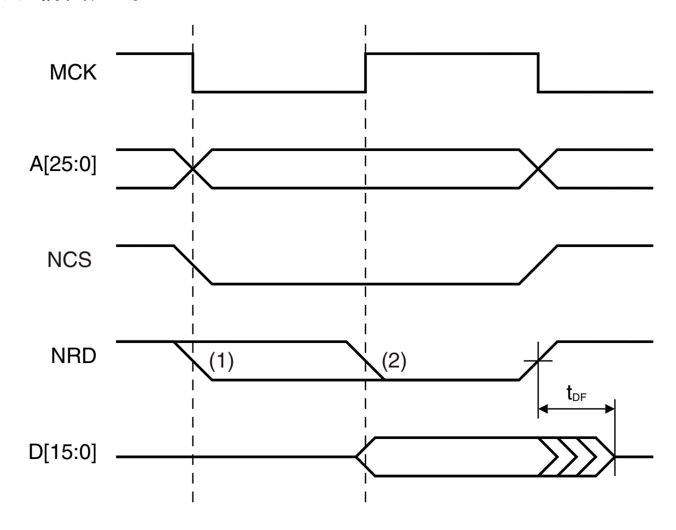
每个外部存储器数据流动输出时间 (t_{DF}) 是由相应的片选 SMC_CSR 寄存器中的 TDF 编程得到的 (见 See “SMC 片选寄存器 ” on page 184.)。TDF 中值表示插入的数据流动等待周期数 (0 ~ 15 间) 及在存储器禁用后允许数据输出到高阻的时间。

时间流动等待状态不会延迟内部存储器访问。因此含长 t_{DF} 的对外部存储器的单访问不会延迟内部存储器程序的执行。

为保证内部存储器系统忙时不被访问，内部访问时 SMC 跟踪编程的内部数据流动时间。

内部存储器访问及对同一个内部存储器的连续读访问不会增加数据流动等待时间。

Figure 54. 时间流动输出延时

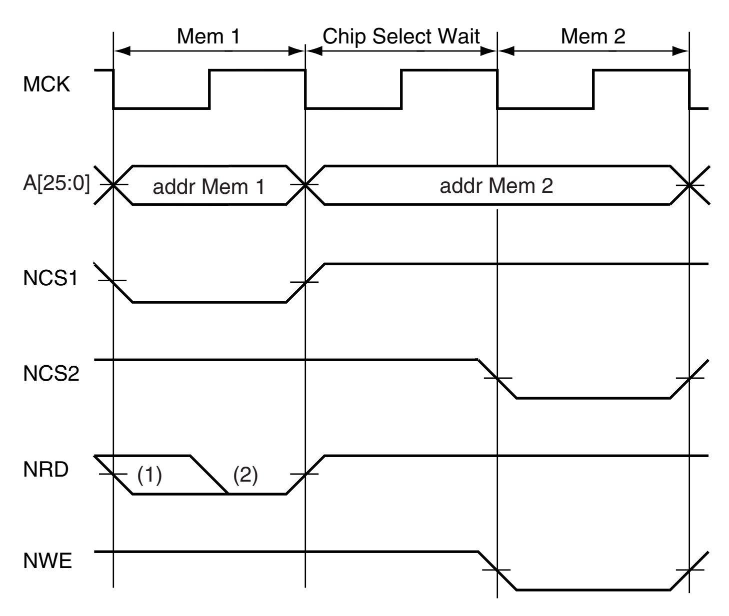


Notes: 1. 预读协议
2. 标准读协议

片选改变等待状态

当连续访问两个不同的外部存储器时自动插入片选等待状态（若没有插入其它类型等待状态）。若已经插入等待状态（例如时间流动等待状态），则不再增加等待状态。

Figure 55. 片选等待状态



Notes: 1. 预读协议
2. 标准读协议

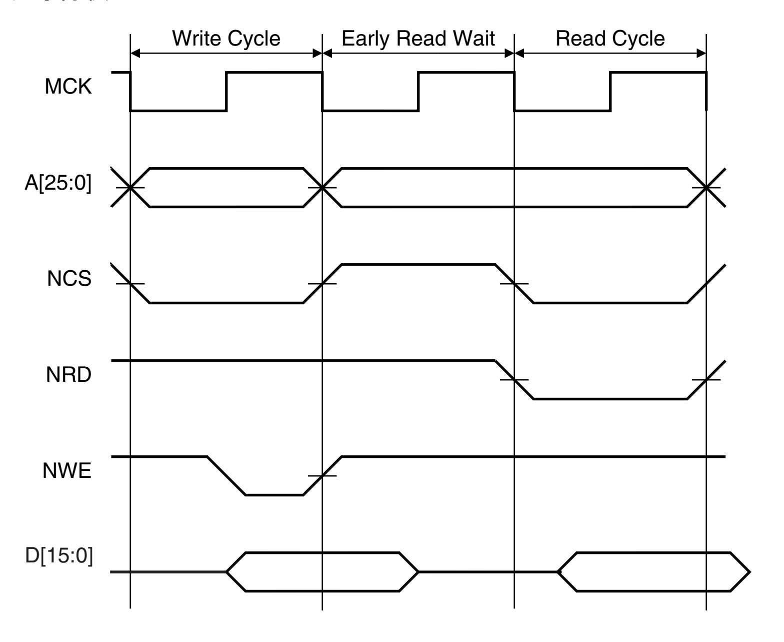
预读等待状态

在预读协议中，当外部写周期后是读周期，将自动插入一个预读等待状态以确保写周期在读周期开始前结束（见 Figure 56)。该等待状态在其它等待状态外产生（即，数据流动等待状态）。

读周期后紧跟一个写周期、相同类型的连续访问或内部与外部存储器间的访问将不会增加等待状态。



Figure 56. 驱动等待状态



启动与保持周期

SMC 允许与不同的启动、保持及脉冲延迟的某些存储器连接。这些参数可编程且定义每部分读写周期的时间。但在预读协议中该特性不能使用。

若 WSEN = 0 (0 标准等待状态) , 编程的启动时间不为零而保持参数等于零，SMC 无法正确工作。

若连续访问两个不同的外部存储器且第二个存储器启动周期已编程，则不会插入片选改变等待状态。 (见 Figure 61 on page 164)。

当第一个存储器数据流动等待状态(t_{DF})已编程而第二个存储器启动周期已编程 ,SMC运行如下:

- 若 t_{DF} 大于或等于启动周期数，插入的启动周期数为 0 (见 Figure 62 on page 164)。
- 若 t_{DF} 小于于启动周期数，插入的 t_{DF} 数为 0 (见 Figure 63 on page 165)。

读访问

读周期可分为启动、脉冲长及保持。启动时间为 1.5 ~ 7.5 个时钟周期间 , 保持时间为 0 ~ 7 时钟周期间，而 脉冲长为 1.5 ~ 128.5 个时钟周期间 ，周期以一递增。

Figure 57. 含启动与保持的读访问

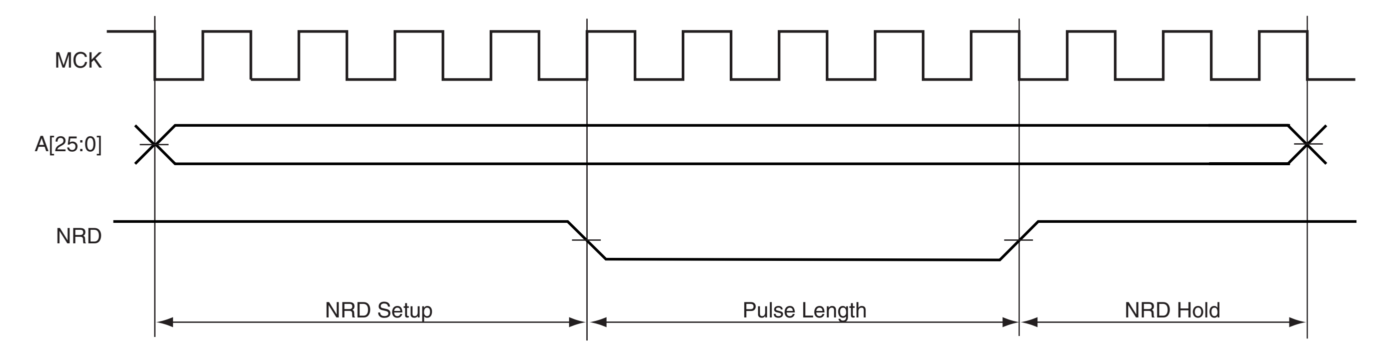
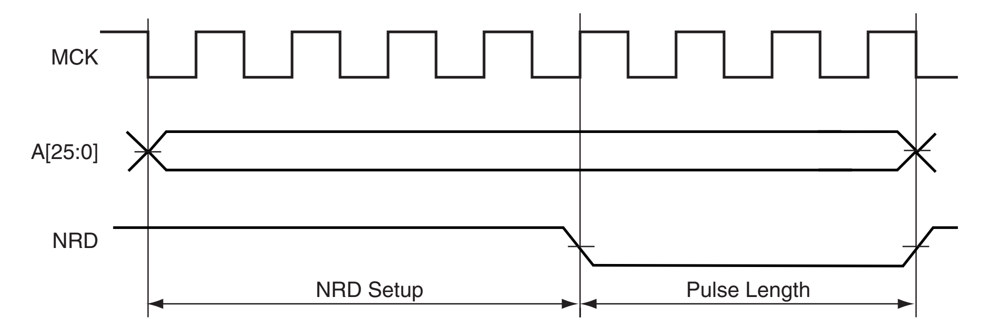


Figure 58. 含启动的读访问



写访问

写周期可分为启动、脉冲长与保持。启动时间为 1.5 ~ 7.5 个时钟周期间，保持时间为 0.5 ~ 7 时钟周期间，而 脉冲长为 1 ~ 128 个时钟周期间 ，周期以一递增。

Figure 59. 含启动与保持的写访问

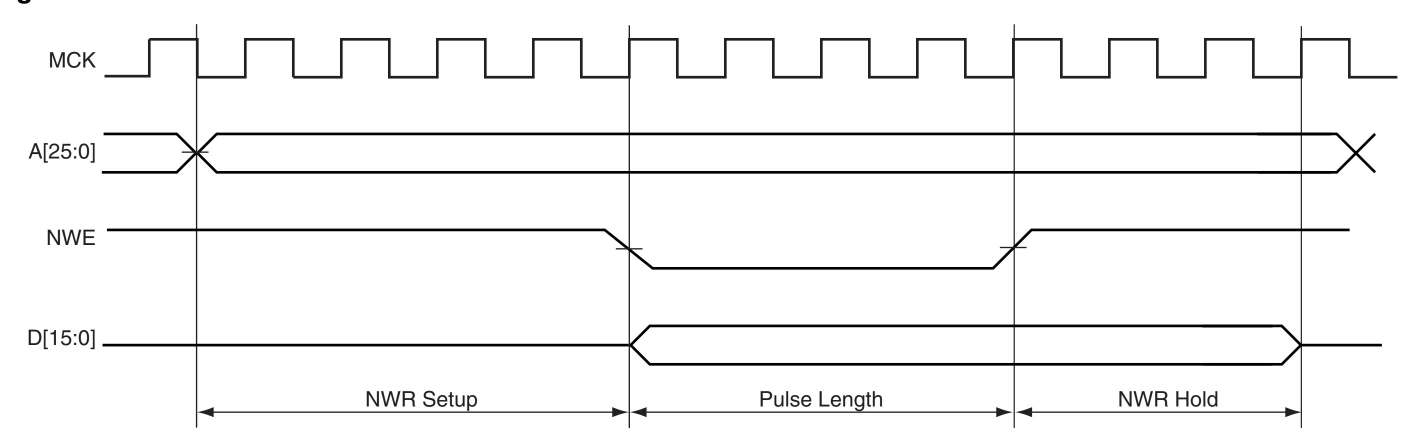
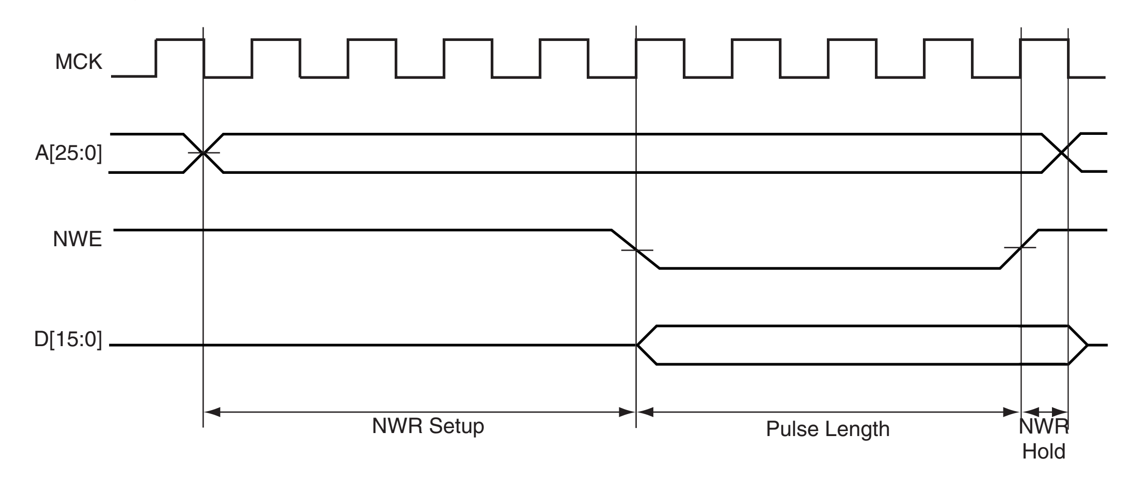


Figure 60. 含启动的写访问



含启动周期的时间流动
等待状态

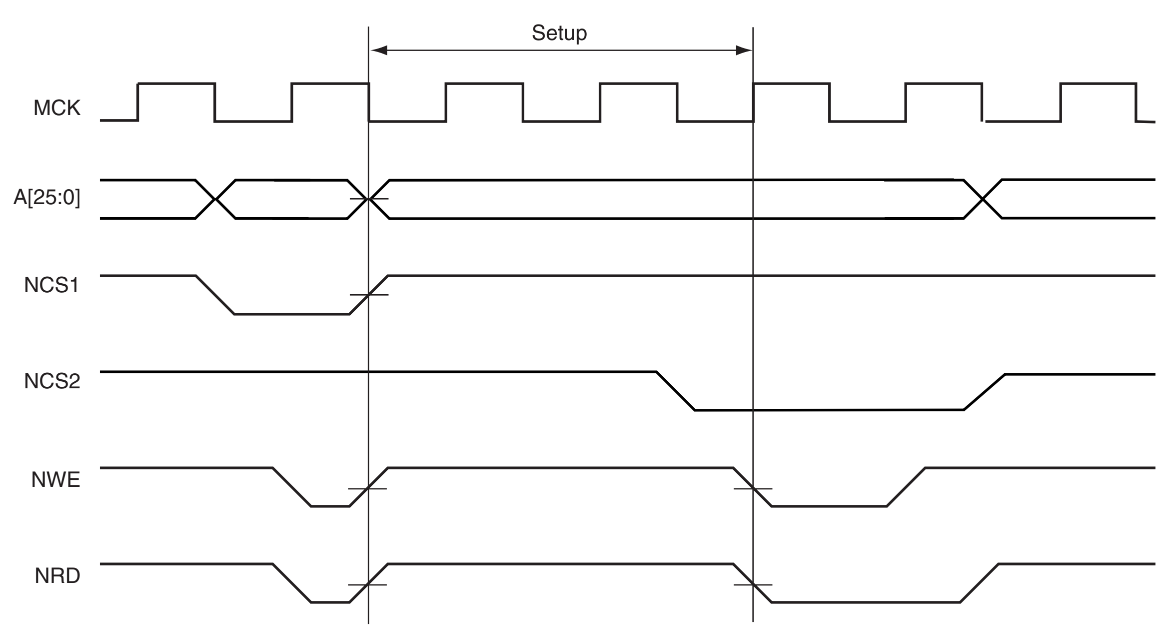


Figure 61. 第二次访问启动编程的连续访问

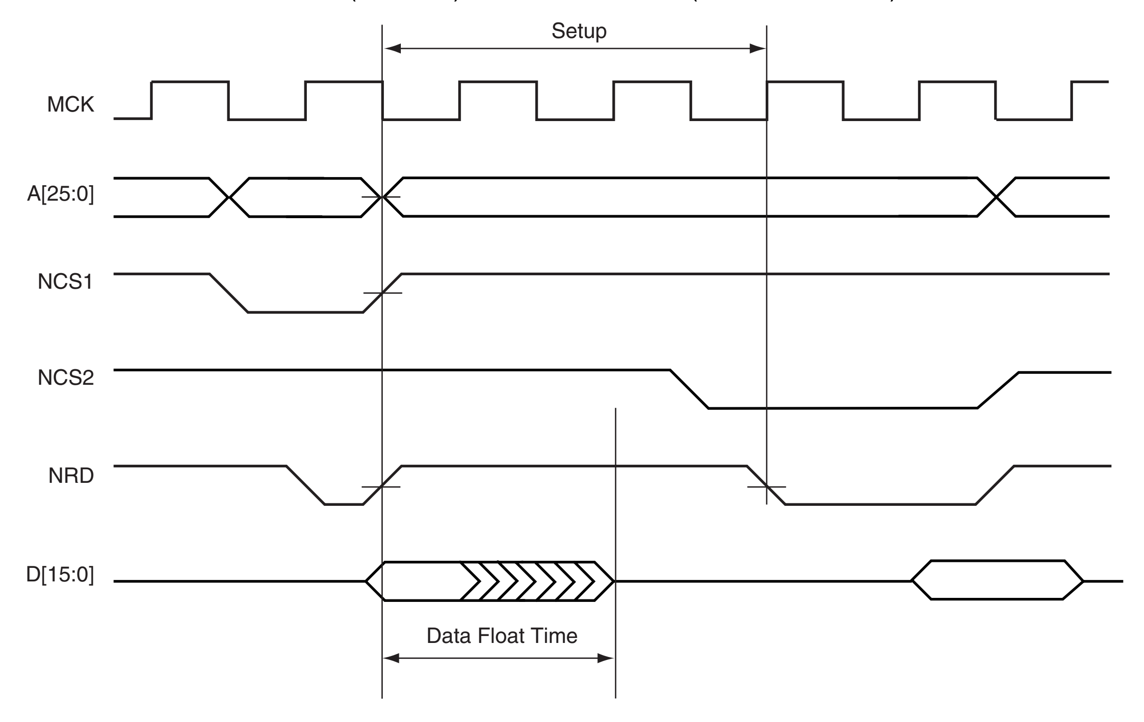
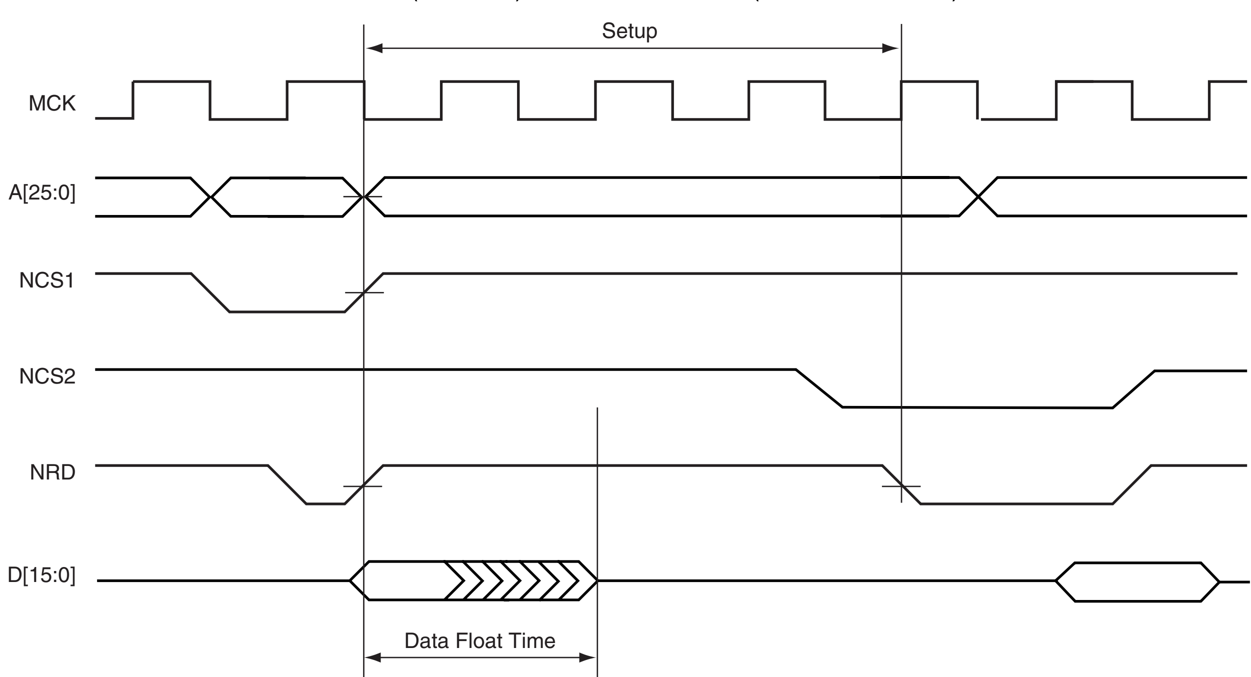


Figure 62. 第一次访问有数据流动等待状态 (TDF = 2) 第二次访问有启动 (NRDSETUP = 1)

Figure 63. 第一次访问有数据流动等待状态 (TDF = 2) 第二次访问有启动 (NRDSETUP = 3)



LCD 接口模式

SMC通过设置SMC_CSR寄存器的ACSS(片选启动地址)位可配置为使用液晶显示(LCD) 控制器工作 (见 See “SMC 片选寄存器 ” on page 184.)。

LCD模式中 ,NCS在前期引导与后期收尾时缩短了1/2/3 个时钟周期 ,以确保有效地址的启动与保持。读访问时，当访问结束时 NCS 上升，数据锁存在 SMC 中。

此外， WSEN 必须置位且 NWS 编程为比 ACSS 多两个以上时钟周期。 LCD 模式下，不推荐使用 RWHOLD 或 RWSETUP。若上述条件无法满足， SMC 无法正确工作。

Figure 64. LCD 接口模式下的读访问

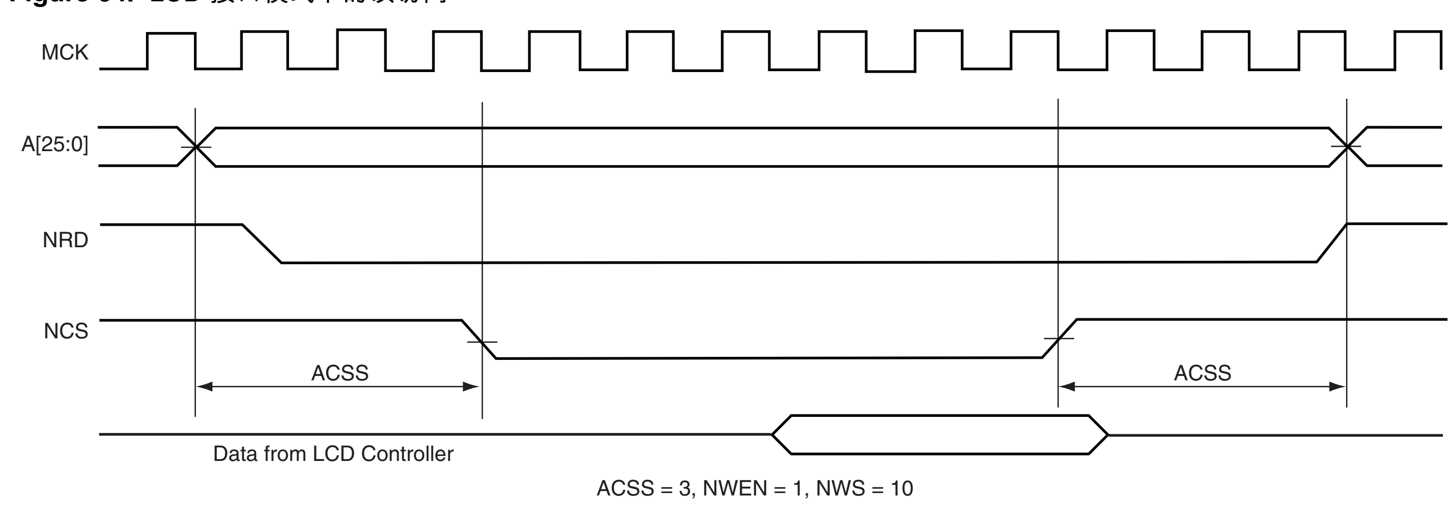
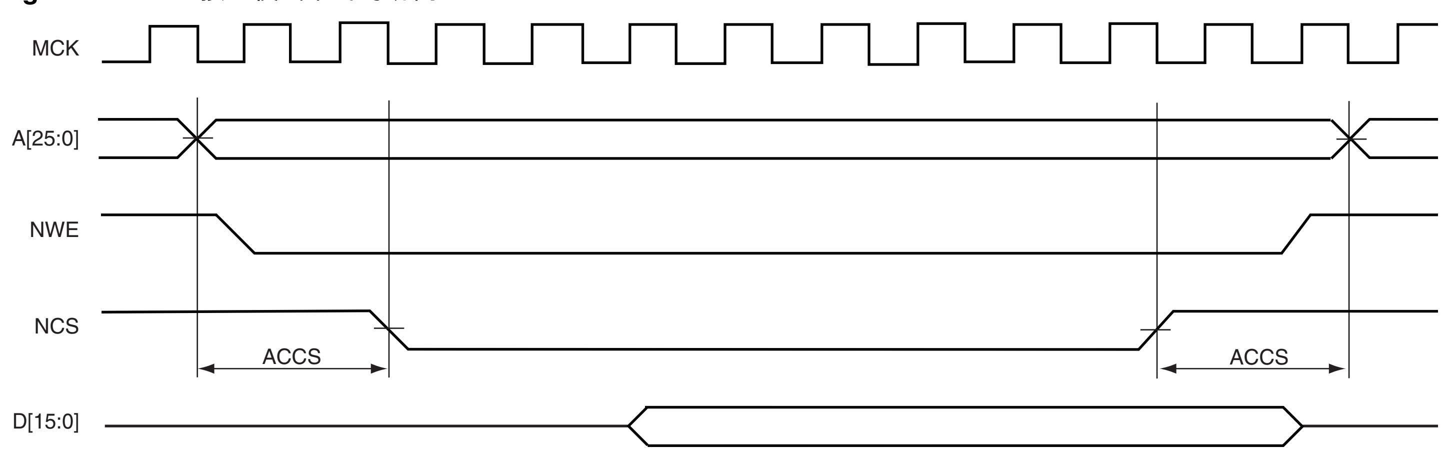


Figure 65. LCD 接口模式下的写访问



存储器访问波形

标准与预读协议下的读访问 Figure 66 on page 167 到 Figure 69 on page 170 给出两种外部存储器读协议的实例。

Figure 66. 不含 t_{DF} 的标准读协议

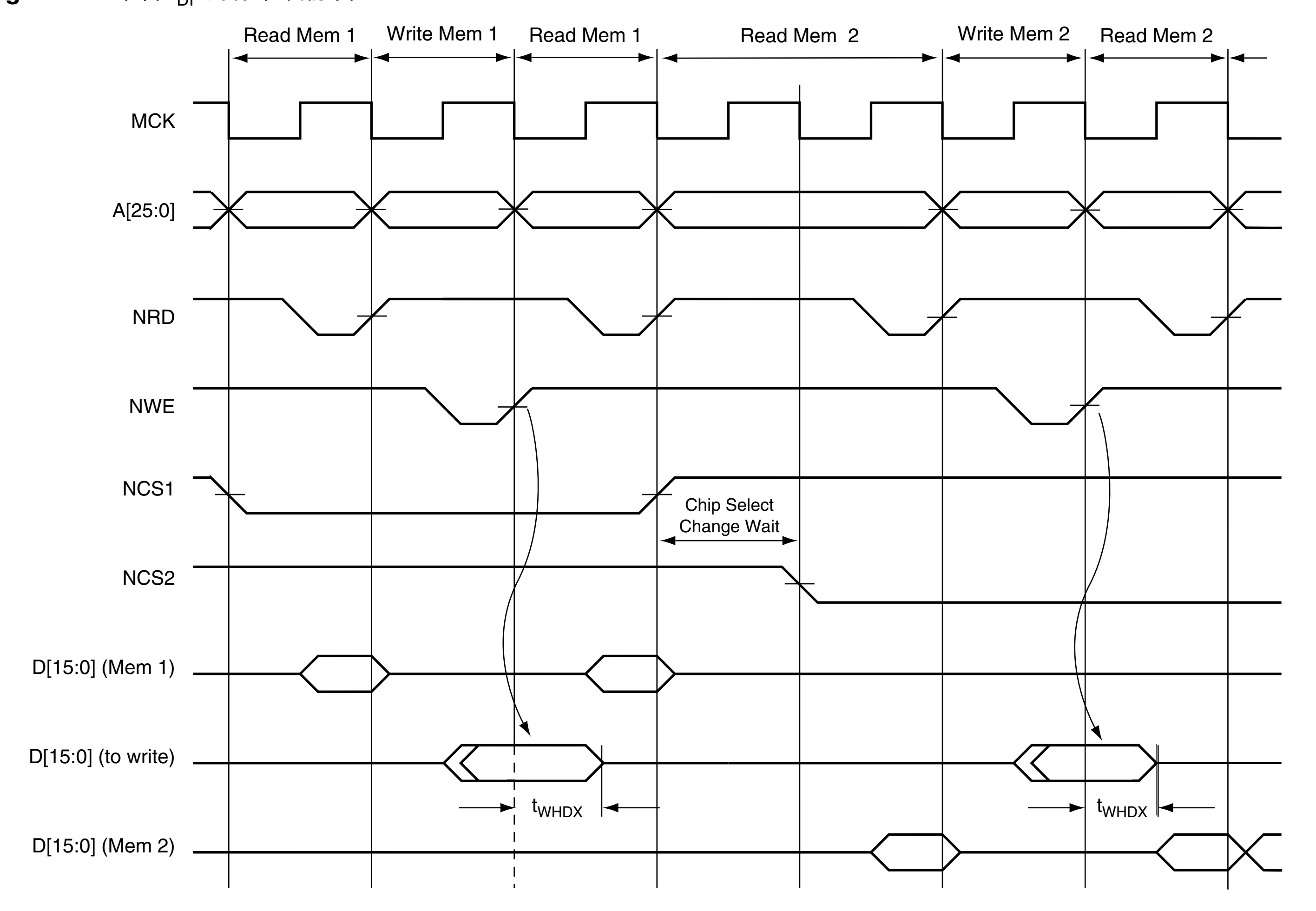


Figure 67. 不含 t_{DF} 的预读协议

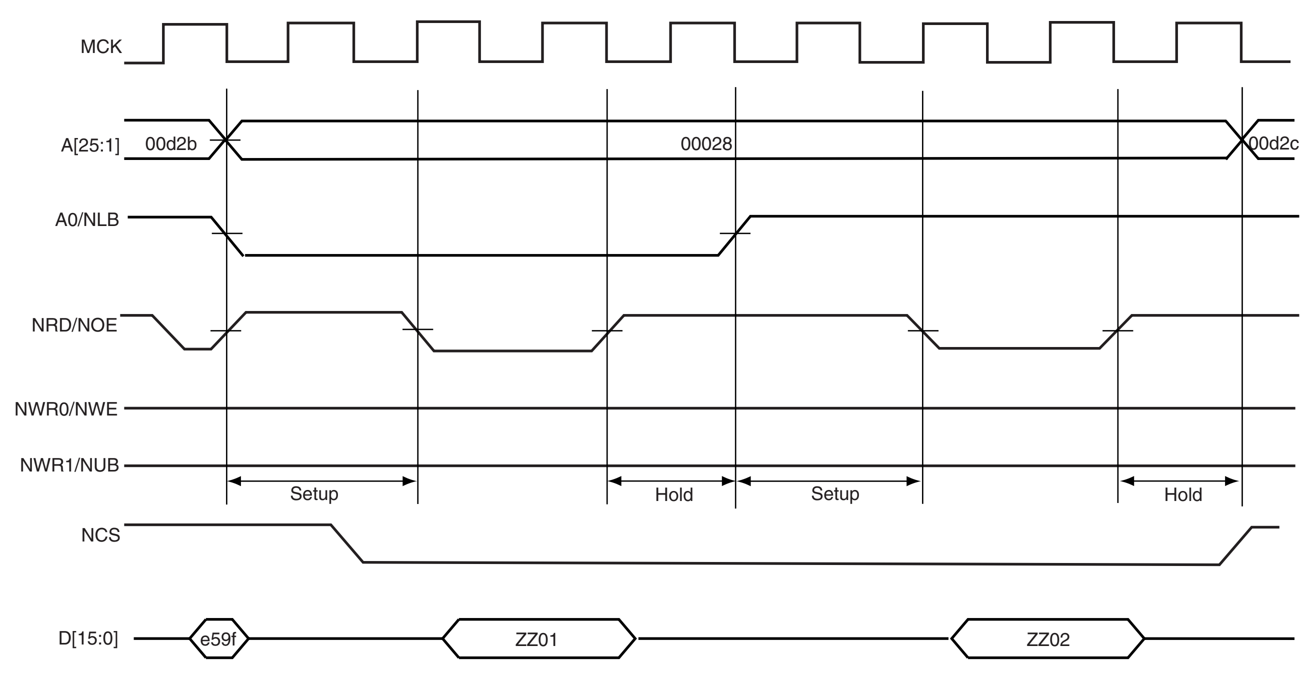
Figure 68. 包含 t_{DF} 的标准读协议

Read Mem 1
Write Mem 1
Read Mem 1
Read Mem 2
Read Mem 2
Write Mem 2
Write Mem 2
Data Float Wait
Data Float Wait
Data Float Wait
MCK
A[25:0]
NRD
NWE
NCS1
NCS2
t_{DF}
(t_{DF} = 1)
t_{DF}
(t_{DF} = 1)
D[15:0] (Mem 1)
D[15:0] (to write)
t_{WHDX}
t_{DF}
(t_{DF} = 2)
D[15:0] (Mem 2)

Figure 69. 包含 t_{DF} 的预读协议

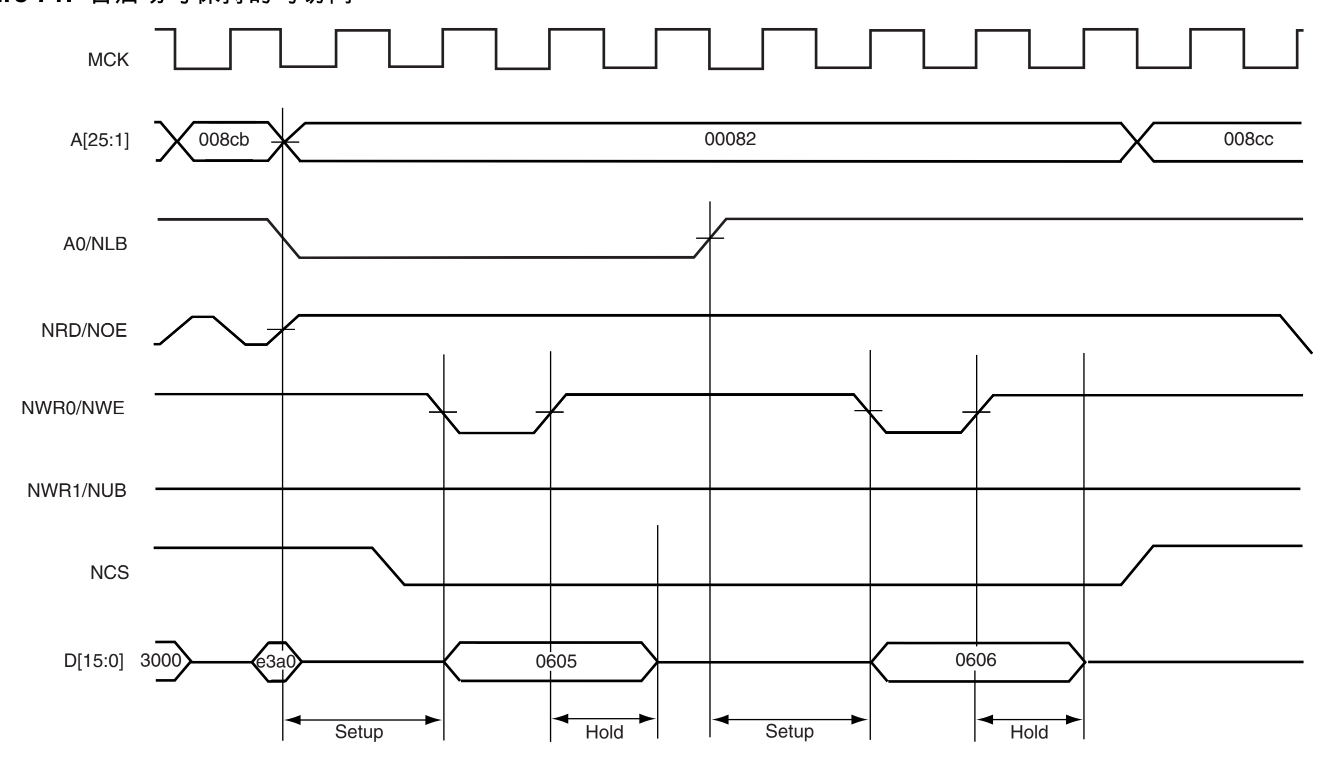
含启动与保持的访问 Figure 70 与 Figure 71 给出含启动与保持周期的读、写访问实例。

Figure 70. 标准读协议下含启动与保持的读访问 (1)



Note: 1. 读访问存储器数据总线宽 = 8， RWSETUP = 1， RWHOLD = 1， WSEN= 1， NWS = 0

Figure 71. 含启动与保持的写访问 (1)



Note: 1. 写访问，存储器数据总线宽 = 8， RWSETUP = 1， RWHOLD = 1， WSEN = 1， NWS = 0



使用 NWAIT 中断信号的访问

Figure 72 on page 172 到 Figure 75 on page 175 给出使用 NWAIT 访问的实例。

Figure 72. 在字节选择类型访问时使用 NWAIT 写访问 [1]

Note: 1. 写访问存储器，数据总线宽 = 16 位， WSEN = 1， NWS = 6

Figure 73. 在字节写类型访问时使用 NWAIT 写访问 (1)

Note: 1. 写访问存储器，数据总线宽 = 16 位， WSEN = 1， NWS = 5



Figure 74. 使用 NWAIT 的写访问 (1)

Chip Select Wait

MCK

NWAIT

NWAIT internally synchronized

A[25:1]

0000033

A0/NLB

NRD/NOE

NWR0/NWE

NWR1/NUB

NCS

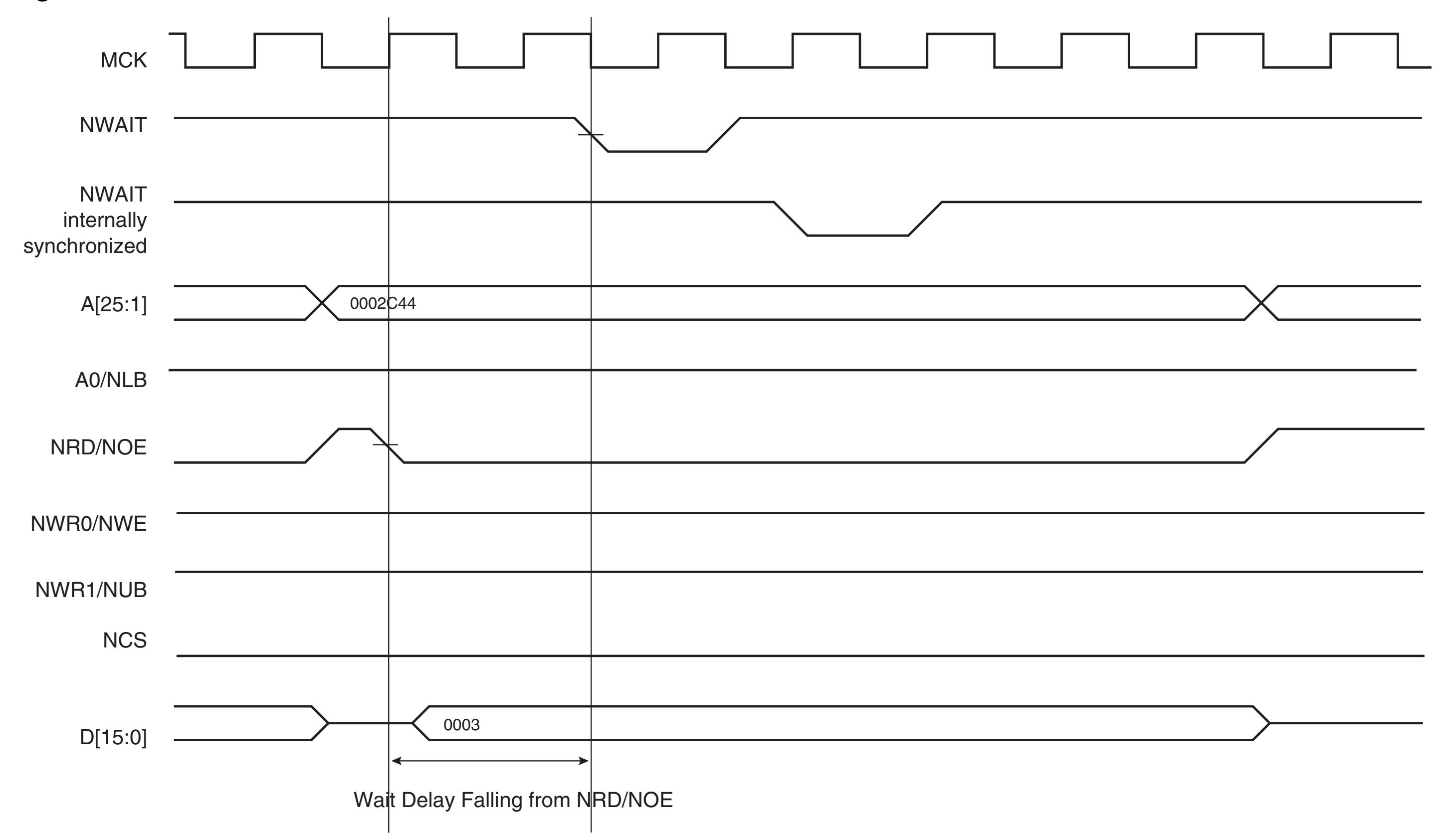
D[15:0]

0403

Wait Delay Falling from NWR0/NWE

Note: 1. 写访问存储器，数据总线宽 = 8 位，WSEN = 1， NWS = 4

Figure 75. 标准协议下使用 NWAIT 的读访问 (1)



Note: 1. 读访问，数据总线宽 = 16， NWS = 5， WSEN = 1

存储器访问实例波形

Figure 76 on page 176 到 Figure 82 on page 182 给出对不同的外部存储器器件读、写访问时的波形。具体配置说明见 Table 46。

Table 46. 存储器访问波形

图序号	等待状态数目	总线宽度	传输数据大小
Figure 76	0	16	字
Figure 77	1	16	字
Figure 78	1	16	半字
Figure 79	0	8	字
Figure 80	1	8	半字
Figure 81	1	8	字节
Figure 82	0	16	字节

Figure 76. 0 等待状态， 16 位总线宽度，字传输

MCK

A[25:1] addr addr+1

NCS

NLB

NUB

Read Access

- Standard Read Protocol

NRD

D[15:0] $B_2 B_1$ $B_4 B_3$

- Early Read Protocol

NRD

D[15:0] $B_2 B_1$ $B_4 B_3$

Write Access

- Byte Write/ Byte Select Option

NWE

D[15:0] $B_2 B_1$ $B_4 B_3$

Figure 77. 1 等待状态， 16 位总线宽度，字传输

1 Wait State
1 Wait State
MCK
A[25:1]
addr
addr+1
NCS
NLB
NUB
Read Access
• Standard Read Protocol
NRD
D[15:0]
B_2 B_1
B_4 B_3
• Early Read Protocol
NRD
D[15:0]
B_2B_1
B_4B_3
Write Access
• Byte Write/
Byte Select Option
NWE
D[15:0]
B_2B_1
B_4B_3

Figure 78. 1 等待状态， 16 位总线宽度，半字传输

1 Wait State

MCK

A[25:1]

NCS

NLB

NUB

Read Access

• Standard Read Protocol

NRD

D[15:0]

$B_2 B_1$

• Early Read Protocol

NRD

D[15:0]

$B_2 B_1$

Write Access

• Byte Write/
Byte Select Option

NWE

D[15:0]

$B_2 B_1$

Figure 79. 0 等待状态，8 位总线宽度，字传输

MCK

A[25:0]

addr addr+1 addr+2 addr+3

NCS

Read Access

・Standard Read Protocol

NRD

D[15:0]

X B1 X B2 X B3 X B4

・Early Read Protocol

NRD

D[15:0]

X B1 X B2 X B3 X B4

Write Access

NWR0

NWR1

D[15:0]

X B1 X B2 X B3 X B4

Figure 80. 1 等待状态，8 位总线宽度，半字传输

1 Wait State

1 Wait State

MCK

A[25:0]

Addr

Addr+1

NCS

Read Access

· Standard Read, Protocol

NRD

D[15:0]

X B1

X B2

· Early Read Protocol

NRD

D[15:0]

X B1

X B2

Write Access

NWR0

NWR1

D[15:0]

X B1

X B2

Figure 81. 1 等待状态 ，8 位总线宽度，字节传输

1 Wait State

MCK

A[25:0]

NCS

Read Access

· Standard Read Protocol

NRD

D[15:0]

XB1

· Early Read Protocol

NRD

D[15:0]

X B1

Write Access

NWR0

NWR1

D[15:0]

X B1



Figure 82. 0 等待状态， 16 位总线宽度，字节传输

MCK

A[25:1] addr X X X 0 addr X X X 0

Internal Address Bus addr X X X 0 addr X X X 1

NCS

NLB

NUB

Read Access

· Standard Read Protocol

NRD

D[15:0] X B_1 B_2X

· Early Read Protocol

NRD

D[15:0] XB_1 B_2X

Write Access

· Byte Write Option

NWR0

NWR1

D[15:0] B_1B_1 B_2B_2

· Byte Select Option

NWE

静态存储控制器 (SMC) 用户接口

静态存储控制器编程使用寄存器见 Table 47。使用 8 个片选寄存器 (SMC_CSR0 ~ SMC_CSR7) 对独立的外部存储器参数编程。

Table 47. 静态存储控制器寄存器映射

偏移	寄存器	名称	访问	复位状态
0x00	SMC 片选寄存器 0	SMC_CSR0	读 / 写	0x00002000
0x04	SMC 片选寄存器 1	SMC_CSR1	读 / 写	0x00002000
0x08	SMC 片选寄存器 2	SMC_CSR2	读 / 写	0x00002000
0x0C	SMC 片选寄存器 3	SMC_CSR3	读 / 写	0x00002000
0x10	SMC 片选寄存器 4	SMC_CSR4	读 / 写	0x00002000
0x14	SMC 片选寄存器 5	SMC_CSR5	读 / 写	0x00002000
0x18	SMC 片选寄存器 6	SMC_CSR6	读 / 写	0x00002000
0x1C	SMC 片选寄存器 7	SMC_CSR7	读 / 写	0x00002000



SMC 片选寄存器

寄存器名称： SMC_CSR0..SMC_CSR7

访问类型： 读 / 写

复位值： 见 Table 47 on page 183

31	30	29	28	27	26	25	24
–	RWHOLD			–	RWSETUP		

23	22	21	20	19	18	17	16
–	–	–	–	–	–	ACSS	

15	14	13	12	11	10	9	8
DRP	DBW		BAT	TDF			

7	6	5	4	3	2	1	0
WSEN	NWS						

- **NWS: 等待状态数目**

该区定义读写信号脉冲长度，从 1 到 128 个周期。

Note: 当 WSEN 为 0，不管前面编程值为何， NWS 将读为 0。

- **WSEN: 等待状态使能**

0：等待状态禁用。

1：等待状态使能。

- **TDF: 数据流动时间**

在 TDF 周期中外部总线屏蔽且不能使用其它片选。最多可定义 15 个周期。

- **BAT: 字节访问类型**

只有当 DBW 定义为 16 或 32 位数据总线时使用。

0：片选线连接两个或四个 8 位宽器件。

1：片选线连接一个 16 位宽器件。

- **DBW: 数据总线宽度**

DBW		数据总线宽度
0	0	保留 (32 位)
0	1	16 位
1	0	8 位
1	1	保留

- **DRP: 数据读协议**

0：标准读协议。

1：预读协议。

- **ACSS: 片选启动地址**

ACSS		片选波形
0	0	标准，访问开始时有效，访问结束失效。

ACSS		片选波形
0	1	访问开始后一周期开始，访问结束时结束。
1	0	访问开始后两周期开始，访问结束时结束。
1	1	访问开始后三周期开始，访问结束时结束。

- **RWSETUP: 读、写信号启动时间**

见下面的定义与说明。

- **RWHOLD: 读、写信号保持时间**

见下面的定义与说明。

RWSETUP			NRD 启动	NWR 启动
0	0	0	½ 周期 (1) 或 0 周期 (2)	½ 周期
0	0	1	1 + ½ 周期	1 + ½ 周期
0	1	0	2 + ½ 周期	2 + ½ 周期
0	1	1	3 + ½ 周期	3 + ½ 周期
1	0	0	4 + ½ 周期	4 + ½ 周期
1	0	1	5 + ½ 周期	5 + ½ 周期
1	1	0	6 + ½ 周期	6 + ½ 周期
1	1	1	7 + ½ 周期	7 + ½ 周期

RWHOLD			NRD 启动	NWR 启动
0	0	0	0	½ 周期
0	0	1	1 周期	1 周期
0	1	0	2 周期	2 周期
0	1	1	3 周期	3 周期
1	0	0	4 周期	4 周期
1	0	1	5 周期	5 周期
1	1	0	6 周期	6 周期
1	1	1	7 周期	7 周期

Notes: 1. 标准读协议下。
2. 预读协议下 (该模式下不能使用 RWSETUP 或 RWHOLD 参数)

NWS(2)	NRD 脉冲长	NWR 脉冲长
0	1 + ½ 周期	1 周期
1	2 + ½ 周期	2 周期
Up to X = 127	X + 1+ ½ 周期	X + 1 周期

Notes: 1. 可参见 “ 启动与保持周期 ” on page 162 及 Figure xx、 Figure yy 与 Figure zz。
2. WSEN 认为是 1。

Figure 83. 读 / 写启动

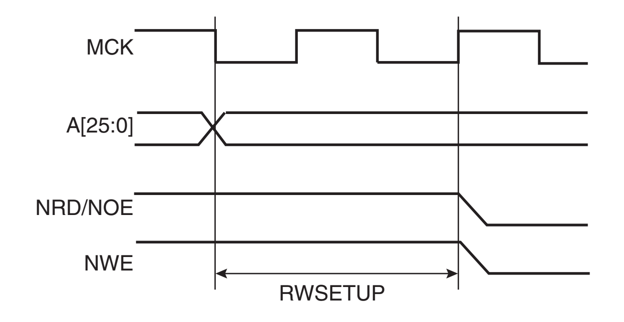


Figure 84. 读保持

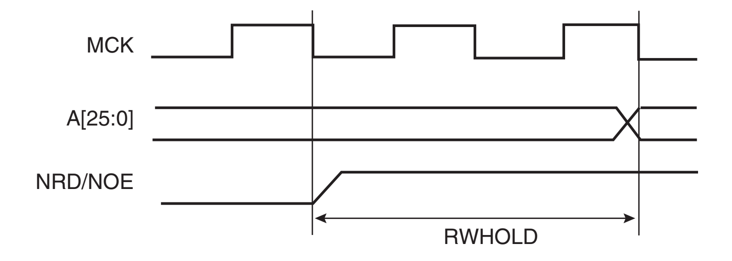
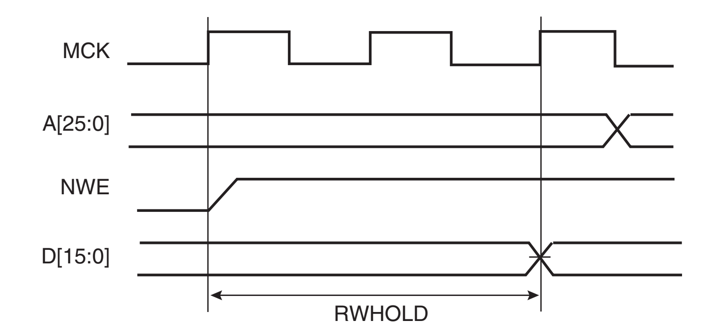


Figure 85. 写保持



SDRAM 控制器 (SDRAMC)

概述

SDRAM 控制器 (SDRAMC) 通过向外部 16 位或 32 位 SDRAM 提供接口来扩展芯片存储能力。支持的页范围从2048到8192，列数由256到2048。它支持字节(8位)，半字(16位)及字(32位)访问。

SDRAM控制器支持对一个位置脉冲长度的读或写。它不支持脉冲的字节读/写或脉冲的半字写。它追踪每组的活跃行，从而最大化 SDRAM 的运行，比如，应用程序应与数据分别放置。为优化运行，建议避免访问相同组的不同行。

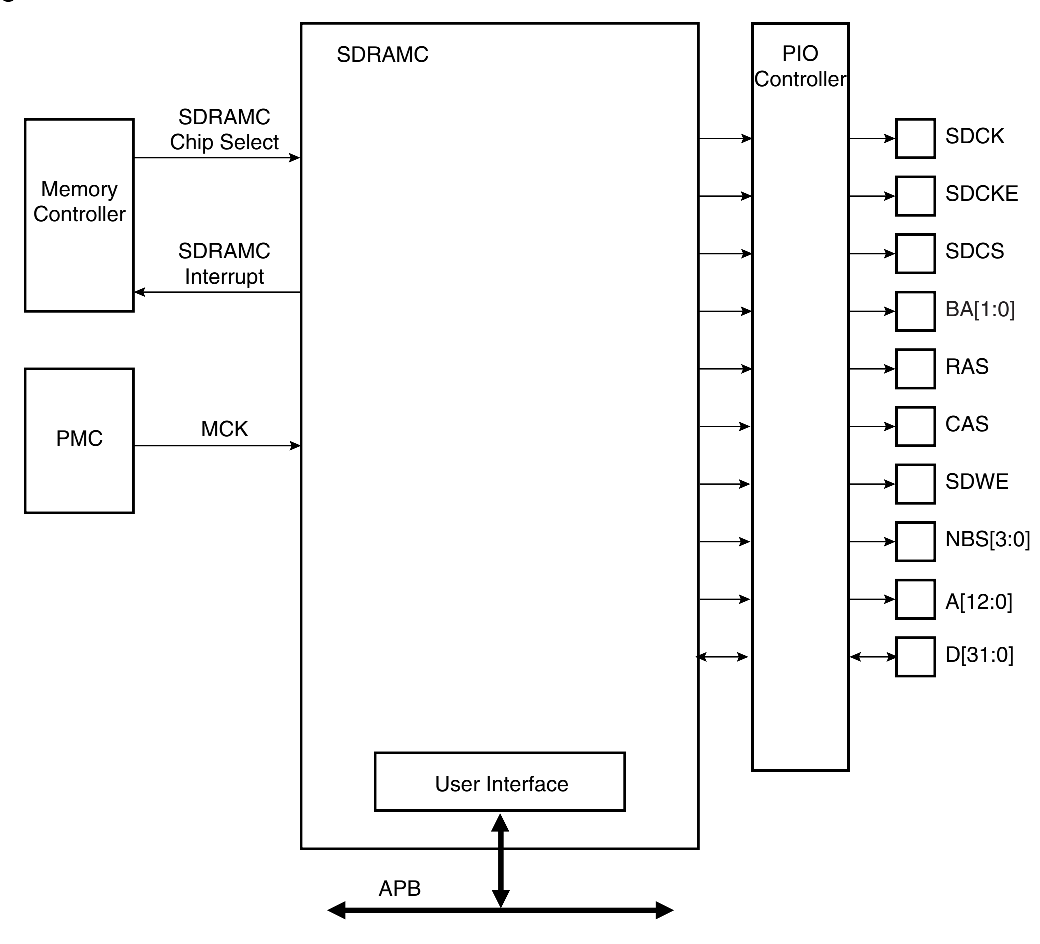
SDRAMC 特性如下：

- 支持许多配置
  - 2K、 4K、 8K 行址存储部分
  - 有两个或四个内部分组的 SDRAM
  - 16 位或 32 位数据通道的 SDRAM
- 编程简便
  - 字、半字、字节访问
  - 到达存储器边界时自动分页
  - 多组 Ping-pong 访问
  - 软件定义的定时参数
  - 自动更新操作，更新频率可编程
- 节能能力
  - 支持自更新与低耗能模式
- 错误检测
  - 更新错误中断
- SDRAM 上电时的软件初始化
- 等待时间设定为 2 个时钟 (CAS 等待时间为 1 个时钟，不支持 3 个时钟)
- 未使用自动预加电命令



方框图

Figure 86. SDRAM 控制器方框图



I/O 口线说明

Table 48. I/O 口线说明

名称	说明	类型	有效电平
SDCK	SDRAM 时钟	输出	
SDCKE	SDRAM 时钟使能	输出	高
SDCS	SDRAM 控制器片选	输出	低
BA[1:0]	分组选择信号	输出	
RAS	行信号	输出	低
CAS	列信号	输出	低
SDWE	SDRAM 写使能	输出	低
NBS[3:0]	数据屏蔽使能信号	输出	低
A[12:0]	地址总线	输出	
D[31:0]	数据总线	I/O	

应用实例

硬件接口

Figure 87给出了使用一个32位宽的数据总线连接SDRAM器件与SDRAM控制器。Figure 88 给出了使用一个 16 位宽的数据总线连接 SDRAM 器件与 SDRAM 控制器。注意 ,这些示例中是器件与 SDRAM 控制器直接连接 ,即没有外部总线接口也不与 PIO 控制器复用。

Figure 87. SDRAM 控制器与 SDRAM 器件连接 :32 位数据总线宽

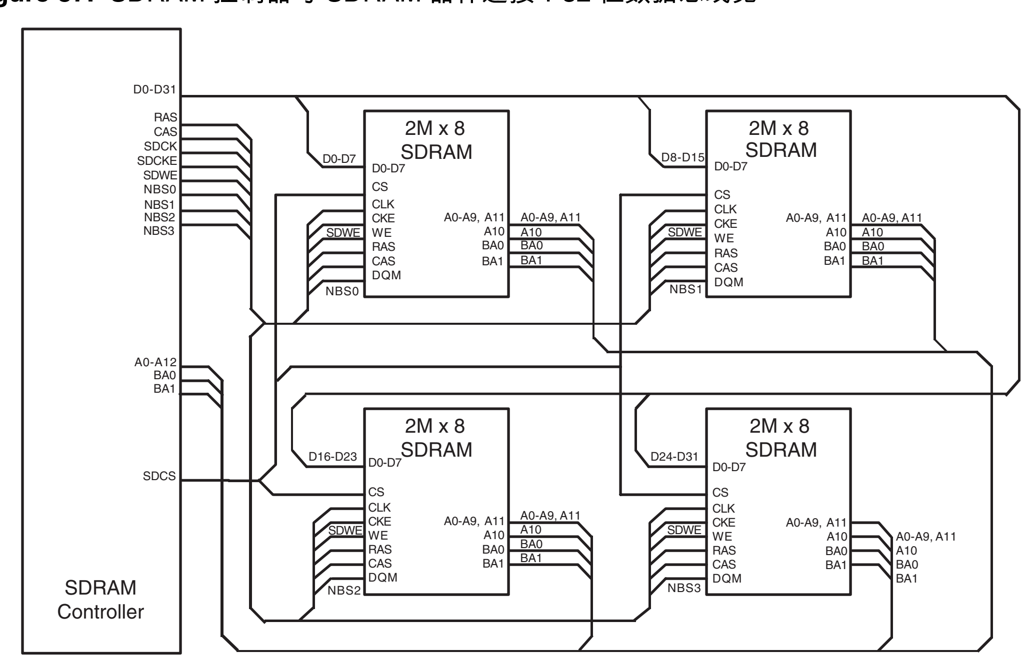
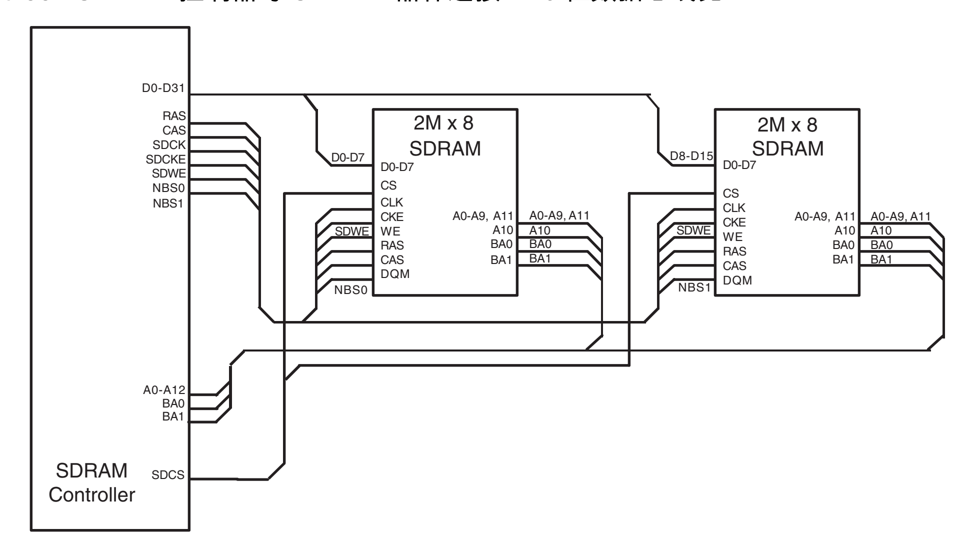


Figure 88. SDRAM 控制器与 SDRAM 器件连接 :16 位数据总线宽



软件接口

SDRAM控制器的功能是使SDRAM器件访问协议对用户透明。Table 49 ~ Table 54给出用户在相关的器件结构中看到的 SDRAM 器件存储映射。不同的配置有说明。



32 位存储器数据总线宽度

Table 49. SDRAM 配置映射：2K 行， 256/512/1024/2048 列

CPU 地址线																											
27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					Bk[1:0]		行 [10:0]											列 [7:0]								M[1:0]	
				Bk[1:0]		行 [10:0]											列 [8:0]									M[1:0]	
			Bk[1:0]		行 [10:0]											列 [9:0]										M[1:0]	
		Bk[1:0]		行 [10:0]											列 [10:0]											M[1:0]	

Table 50. SDRAM 配置映射：4K 行， 256/512/1024/2048 列

CPU 地址线																											
27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				Bk[1:0]		行 [11:0]												列 [7:0]								M[1:0]	
			Bk[1:0]		行 [11:0]												列 [8:0]									M[1:0]	
		Bk[1:0]		行 [11:0]												列 [9:0]										M[1:0]	
	Bk[1:0]		行 [11:0]												列 [10:0]											M[1:0]	

Table 51. SDRAM 配置映射：8K 行， 256/512/1024/2048 列

CPU 地址线																											
27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
			Bk[1:0]		行 [12:0]													列 [7:0]								M[1:0]	
		Bk[1:0]		行 [12:0]													列 [8:0]									M[1:0]	
	Bk[1:0]		行 [12:0]													列 [9:0]										M[1:0]	
Bk[1:0]		行 [12:0]													列 [10:0]											M[1:0]	

Notes: 1. M[1:0] 是 32 位字中的地址字节。
2. Bk[1] = BA1， Bk[0] = BA0。

16 位存储器数据总线宽度

Table 52. SDRAM 配置映射：2K 行， 256/512/1024/2048 列

CPU 地址线																											
27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						Bk[1:0]		行 [10:0]											列 [7:0]								M0
					Bk[1:0]		行 [10:0]											列 [8:0]									M0
				Bk[1:0]		行 [10:0]											列 [9:0]										M0
			Bk[1:0]		行 [10:0]											列 [10:0]											M0

Table 53. SDRAM 配置映射：4K 行， 256/512/1024/2048 列

CPU 地址线																											
27	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
					Bk[1:0]		行 [11:0]												列 [7:0]								M0
				Bk[1:0]		行 [11:0]												列 [8:0]									M0
			Bk[1:0]		行 [11:0]												列 [9:0]										M0
		Bk[1:0]		行 [11:0]												列 [10:0]											M0

Table 54. SDRAM 配置映射：8K 行， 256/512/1024/2048 列

CPU 地址线																											
27	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
				Bk[1:0]		行 [12:0]													列 [7:0]								M0
			Bk[1:0]		行 [12:0]													列 [8:0]									M0
		Bk[1:0]		行 [12:0]													列 [9:0]										M0
	Bk[1:0]		行 [12:0]													列 [10:0]											M0

Notes: 1. M0 为 16 位半字中的地址字节。
2. Bk[1] = BA1， Bk[0] = BA0。

相关产品

SDRAM 器件初始化

初始化序列由软件产生。 SDRAM 器件初始化序列如下：

1. 开始前最少等待 200 µs ；
2. 给 SDRAM 器件送出所有组预充电命令；
3. 提供 8 个自动更新 (CBR) 周期；
4. 一个模式寄存器设置 (MRS) 周期用以对 SDRAM 器件参数编程 , 特别是 CAS 等待时间与脉冲长度；
5. t_{MRD} 后 3 个时钟周期执行一个正常模式命令；
6. 在 SDRAMC 更新定时寄存器的计数域中写入更新速率 (更新速率 = 更新循环间延迟)。

经过以上六步后， SDRAM 器件可完全使用其功能。

SDRAMC 模式寄存器中的命令域产生命令 (NOP、 MRS、 CBR、正常模式) 。

Figure 89. SDRAM 器件初始化序列

I/O 口线

用于连接 SDRAM 控制器的引脚可与 PIO 口线复用。必须先对 PIO 编程分配 SDRAM 控制器引脚的外设功能。若应用程序中未使用 SDRAM 控制器的 I/O 口线，它们可以被 PIO 控制器用于其它目的。

中断

SDRAM控制器中断(更新错误通知)与存储控制器连接。该中断可与其它系统外设中断相或，最终作为系统中断源 (源 1) 提供给 AIC (高级中断控制器)。

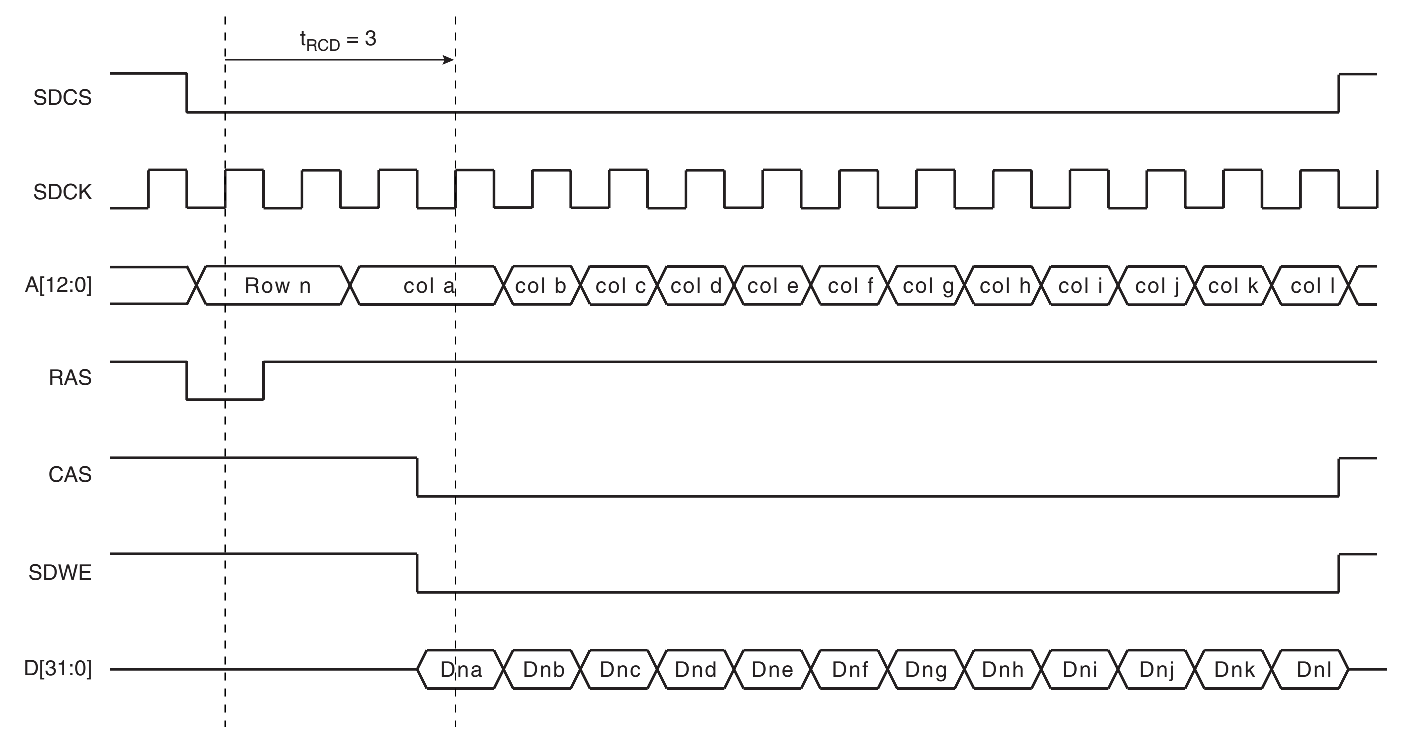
使用 SDRAM 控制器中断请求时要先对 AIC 编程。

功能说明

SDRAM 控制器写周期

SDRAM 控制器允许脉冲访问或单个访问。SDRAM 控制器使用主机请求访问的传输类型信号初始化脉冲访问。若下面是连续写访问，向 SDRAM 器件写入；若当前访问到页边界，或下一个写访问的是另一行， SDRAM 控制器产生一个预充电命令，激活新行并初始化写命令。为遵守SDRAM定时参数，在预充电/激活(t_{RP})命令与激活/写(t_{RCD})命令间插入额外的时钟周期。对于这些定时参数的定义见 "SDRAMC 配置寄存器 " on page 202。如 Figure 90 所示。

Figure 90. 脉冲写， 32 位 SDRAM 访问

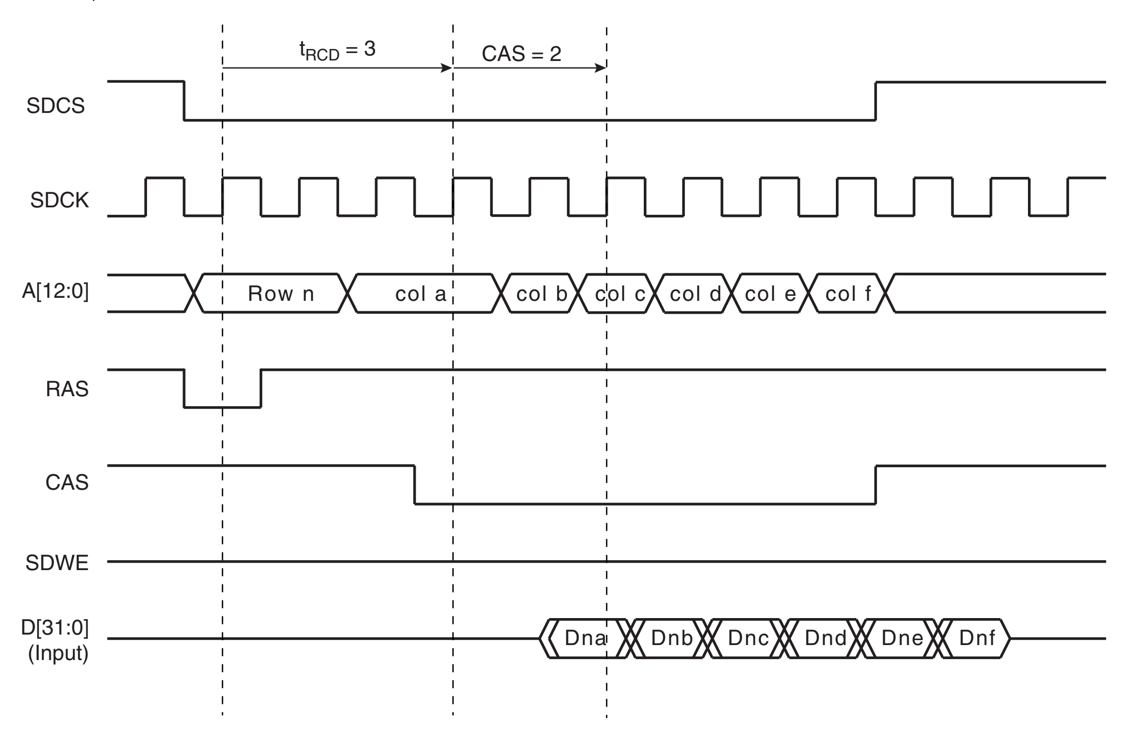




SDRAM 控制器读周期

SDRAM 控制器允许脉冲访问或单个访问。SDRAM 控制器使用主机请求访问的传输类型信号初始化脉冲访问。若下面是连续读访问，由 SDRAM 器件读出；若当前访问到页边界，或下一个读访问的是另一行， SDRAM 控制器产生一个预充电命令，激活新行并初始化读命令。为遵守 SDRAM定时参数，在预充电/激活(t_{RP})命令与激活/读(t_{RCD})命令间插入额外的时钟周期。读命令后，为适用 cas 延迟， SDRAM 将产生两个 cas 延迟。对于这些定时参数的定义见 “SDRAMC 配置寄存器 ” on page 202。如 Figure 91 所示。

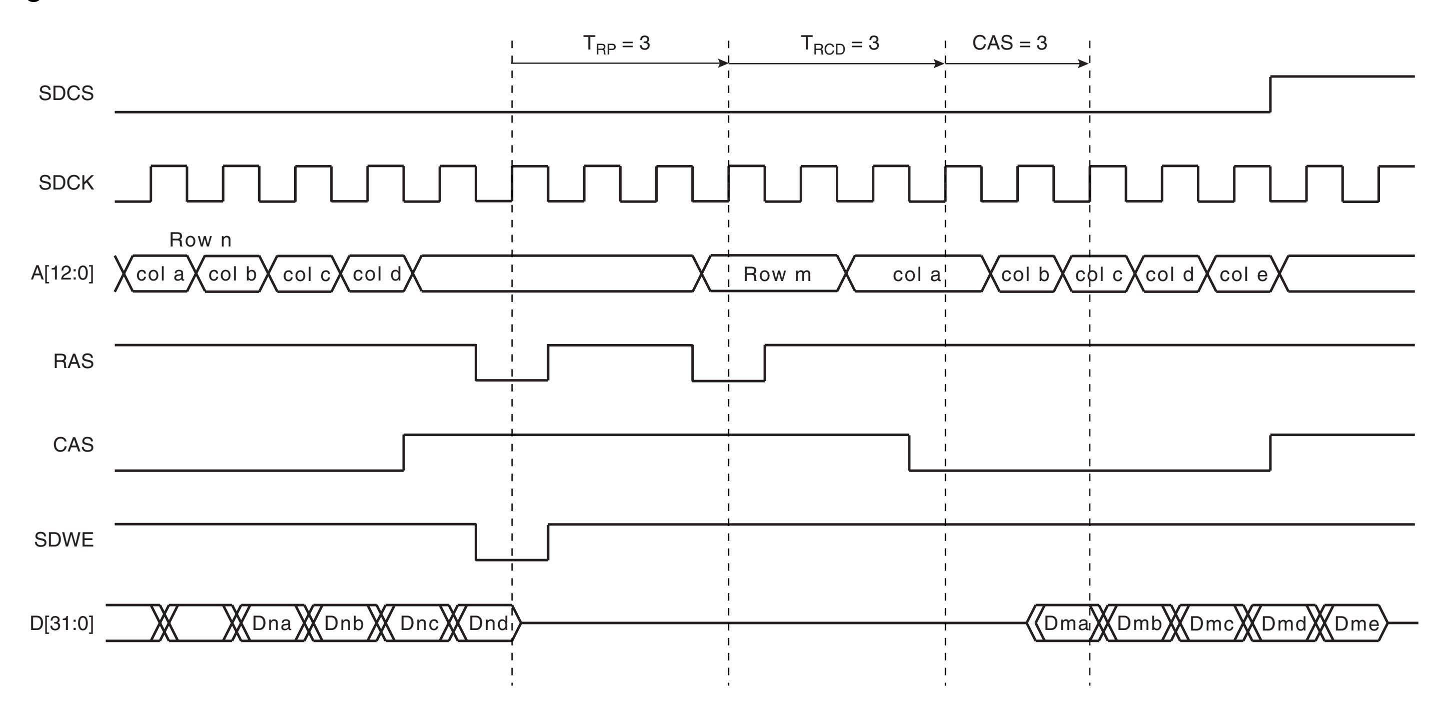
Figure 91. 脉冲读， 32 位 SDRAM 访问



边界管理

当到达存储器行边界时，自动插入一个页面分割。此时，SDRAM 控制器产生一个预充电命令，激活新行并初始化读或写命令。为遵守 SDRAM 定时参数，在预充电 / 激活 (t_{RP}) 命令与激活 / 读 (t_{RCD}) 命令间插入额外的时钟周期，见 Figure 92 的说明。

Figure 92. 读脉冲的边界行访问



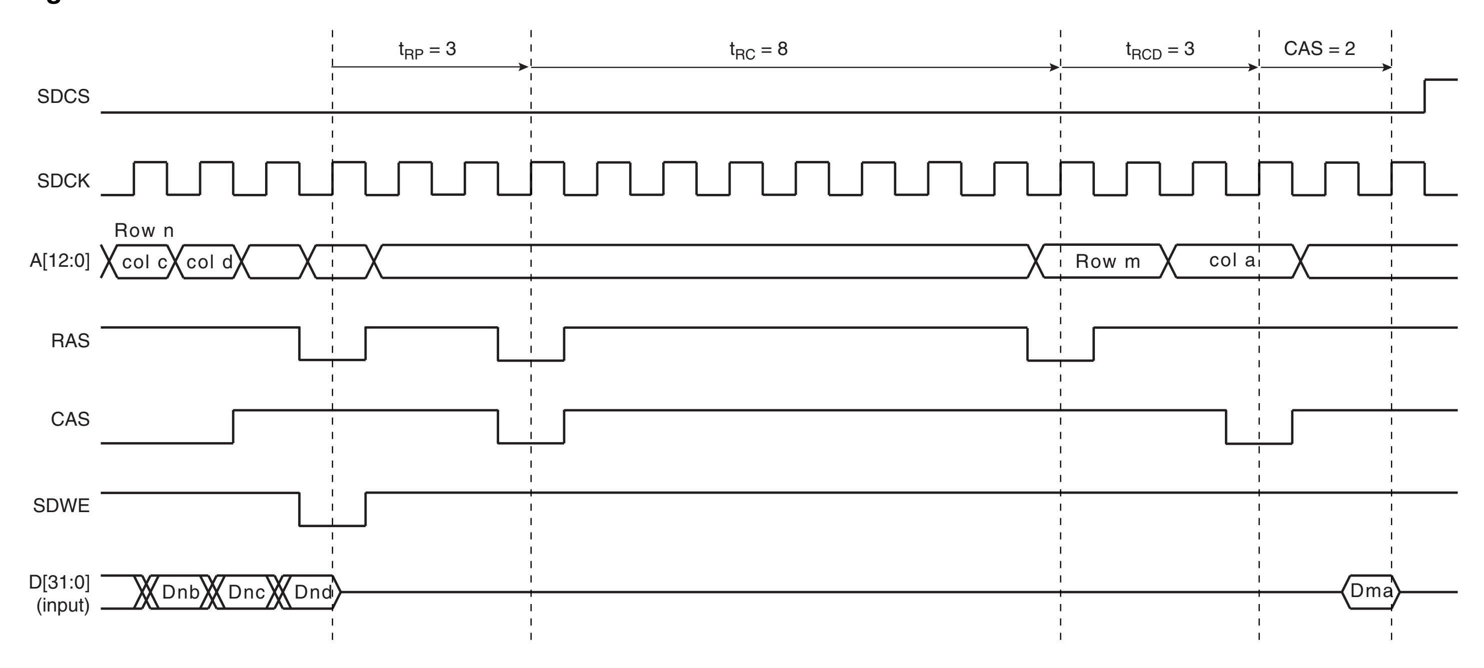
SDRAM 控制器更新周期

自动更新命令用以刷新 SDRAM 器件。刷新地址由 SDRAM 产生且每次自动刷新后自加一。SDRAM控制器周期性产生这些自动刷新命令。定时器载入SDRAMC_TR寄存器中值，用以说明在刷新周期间的时钟周期数。

当前一个自动刷新命令没有执行将产生一个刷新误差中断。通过读中断状态寄存器 (SDRAMC_ISR) 可以确认。

当 SDRAM 控制器初始化一个 SDRAM 器件刷新时，内部存储器访问不会延时。但若 CPU 要访问 SDRAM，从机将显示器件忙并插入 ARM BWAIT 信号，见 Figure 93。

Figure 93. 刷新周期后进行读访问



电源管理

自刷新模式

自刷新模式用于休眠模式下，即没有对 SDRAM 器件访问。此时功耗极低。该模式通过对 SDRAMC_SRR 中的自刷新命令编程来激活。自刷新模式下 ,SDRAM 器件不用外部时钟使用自己的内部时钟来保持数据，从而执行自己的自动刷新周期。除 SDCKE 保持低电平时，SDRAM 器件输入均不必在意。一旦选中 SDRAM 器件， SDRAM 控制器提供一个命令序列并退出自刷新模式，以禁用自刷新命令位。

为重新激活该模式，自刷新命令位必须重新编程。

SDRAM 器件保持为自刷新模式时间不得少于 t_{RAS} 或保持自刷新模式在不定的时间段，如图 Figure 94 所示。

Figure 94. 自刷新模式动作

Self Refresh Mode
T_{XSR} = 3
SRCB = 1
Write SDRAMC_SRR
A[12:0]
Row
SDCK
SDCKE
SDCS
RAS
CAS
SDWE
Access Request
to the SDRAM Controller



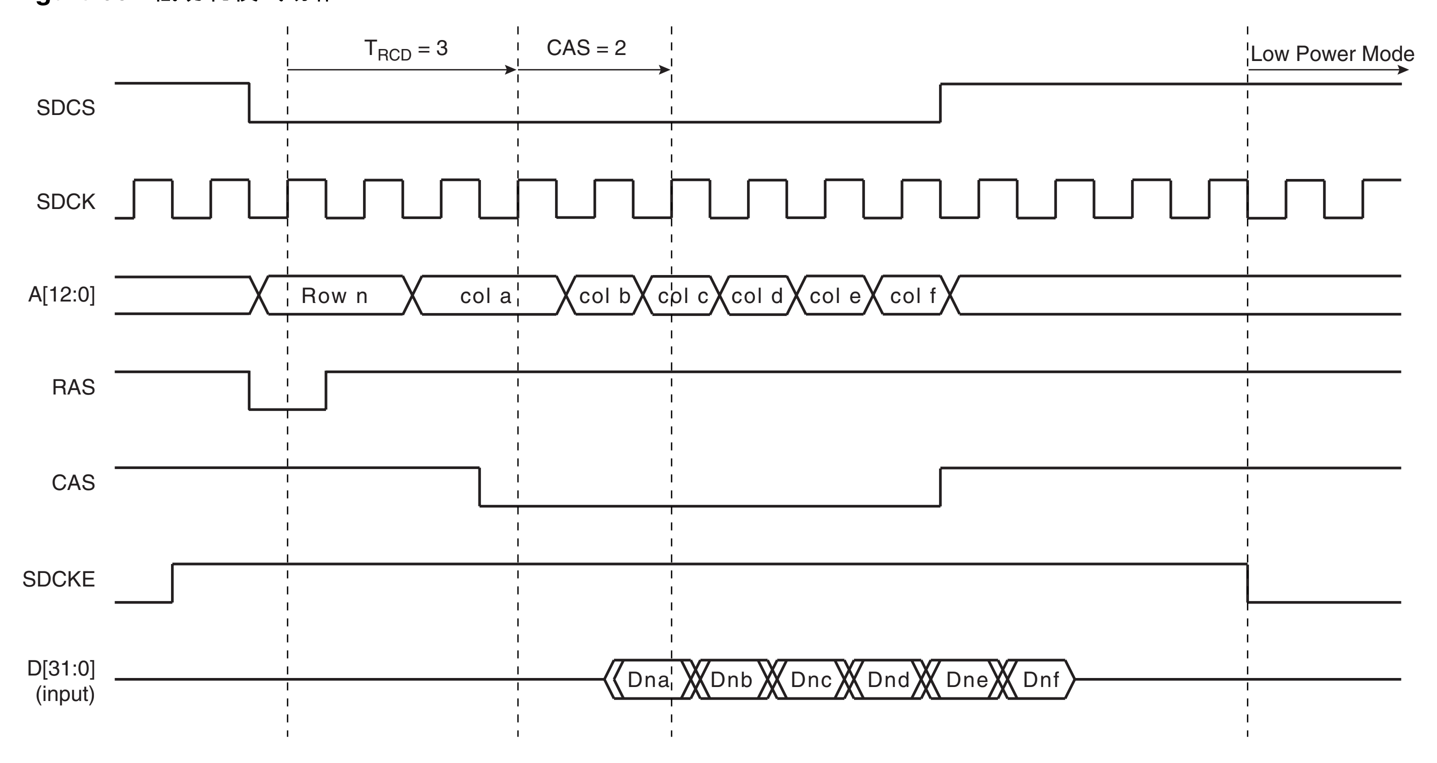
低功耗模式

低功耗模式用于休眠模式下，即没有对 SDRAM 器件访问。该模式下功耗大于自刷新模式。其状态与正常模式类似（非低功耗模式 / 非自刷新模式），当 SDRAM 器件不能访问后， SDCKE 引脚为低且输入与输出缓冲器失效。与自刷新模式相反，低功耗模式下 SDRAM 保持时间低于刷新周期（完整的器件重刷新操作需 64 ms)。由于该模式下不执行自动刷新操作， SDRAM 控制器执行刷新操作。为退出低功耗模式，需要一个 NOP 命令。退出速度比在自刷新模式下快。

当自刷新模式使能时，建议不要使能低功耗模式。当低功耗模式使能时，建议不要使能自刷新模式。

见 Figure 95。

Figure 95. 低功耗模式动作



SDRAM 控制器 (SDRAMC) 用户接口

Table 55. SDRAM 控制存储器映射

偏移	寄存器	名称	访问	复位状态
0x00	SDRAMC 模式寄存器	SDRAMC_MR	读 / 写	0x00000010
0x04	SDRAMC 刷新定时器寄存器	SDRAMC_TR	读 / 写	0x00000800
0x08	SDRAMC 配置寄存器	SDRAMC_CR	读 / 写	0x2A99C140
0x0C	SDRAMC 自刷新寄存器	SDRAMC_SRR	只写	–
0x10	SDRAMC 低功耗寄存器	SDRAMC_LPR	读 / 写	0x0
0x14	SDRAMC 中断使能寄存器	SDRAMC_IER	只写	–
0x18	SDRAMC 中断禁用寄存器	SDRAMC_IDR	只写	–
0x1C	SDRAMC 中断屏蔽寄存器	SDRAMC_IMR	只读	0x0
0x20	SDRAMC 中断状态寄存器	SDRAMC_ISR	只读	0x0



SDRAMC 模式寄存器

寄存器名称： SDRAMC_MR

访问类型： 读 / 写

复位值： 0x00000010

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	DBW	MODE			

- **MODE: SDRAMC 命令模式**

该域定义了当访问 SDRAM 器件时 SDRAM 控制器命令。

模式				说明
0	0	0	0	正常模式，正常解码对 SDRAM 的访问。
0	0	0	1	当访问 SDRAM 器件时， SDRAM 控制器发出一个 NOP 命令。
0	0	1	0	当访问 SDRAM 器件时， SDRAM 控制器发出一个 " 所有组预充电 " 命令。
0	0	1	1	当访问 SDRAM 器件时， SDRAM 控制器发出一个 " 载入模式寄存器 " 命令。与 SDRAM 器件基地址相关的偏移地址用于对模式寄存器编程。例如，当模式激活，对 "SDRAM_Base + offset" 地址访问产生一个 " 载入模式寄存器 " 命令 ，将 " 偏移值 " 写入 SDRAM 器件模式寄存器。
0	1	0	0	当访问 SDRAM 器件时， SDRAM 控制器发出一个 " 刷新 " 命令。此前必须先发出一个 " 所有组预充电 " 命令。

- **DBW: 数据总线宽度**

0：数据总线宽度为 32 位。

1：数据总线宽度为 16 位。

SDRAMC 刷新定时器寄存器

寄存器名称： SDRAMC_TR
访问类型： 读 / 写
复位值： 0x00000800

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	COUNT			

7	6	5	4	3	2	1	0
COUNT							

• **COUNT: SDRAMC 刷新定时器寄存器**

12 位域载入定时器中产生刷新脉冲。每个刷新脉冲产生，初始化脉冲刷新。载入值由 SDRAMC 时钟频率决定 (MCK：主机时钟)， SDRAM 器件刷新速率与脉冲刷新长的典型值为 15.6 µs/ 行。

即使复位值不为 0，为刷新 SDRAM 器件，该 12 位域必须写入。若不满足该条件，将不会发出刷新命令， SDRAM 器件也不会执行刷新操作。



SDRAMC 配置寄存器

寄存器名称： SDRAMC_CR

访问类型： 读 / 写

复位值： 0x2A99C140

31	30	29	28	27	26	25	24
–	TXSR				TRAS		

23	22	21	20	19	18	17	16
TRAS	TRCD				TRP		

15	14	13	12	11	10	9	8
TRP	TRC				TWR		

7	6	5	4	3	2	1	0
TWR	CAS		NB	NR		NC	

- **NC: 列位数**

复位值为 8 列。

NC		列位数
0	0	8
0	1	9
1	0	10
1	1	11

- **NR: 行位数**

复位值是 11 行位。

NR		行位数
0	0	11
0	1	12
1	0	13
1	1	保留

- **NB: 分组数**

复位值为 2 个分组。

NB	分组数
0	2
1	4

- **CAS: CAS 延迟**

复位值为 2 周期。

在 SDRAMC 中，仅管理 2 周期 CAS 延迟。若要得到其它值，必须编程。

CAS		CAS 延迟 (周期)
0	0	保留
0	1	保留
1	0	2
1	1	保留

- **TWR: 写恢复延迟**

复位值为 2 周期。

该域定义写恢复的周期数，值在 2 ~ 15 间。

若 TWR 不大于 2，默认插入 2 周期。

- **TRC: 行周期延迟**

复位值为 8 周期。

该域定义刷新与激活命令间的延迟数，值在 2 ~ 15 间。

若 TRC 不大于 2，默认插入 2 周期。

- **TRP: 行预充电延迟**

复位值为 3 周期。

该域定义预充电与其它命令间的延迟数，值在 2 ~ 15 间。

若 TRP 不大于 2，默认插入 2 周期。

- **TRCD: 行到列延迟**

复位值为 3 周期。

该域定义激活与读 / 写命令间的延迟数，值在 2 ~ 15 间。

若 TRCD 不大于 2，默认插入 2 周期。

- **TRAS: 激活到预充电延迟**

复位值为 5 周期。

该域定义激活与预充电命令间的延迟数，值在 2 ~ 15 间。

若 TRAS 不大于 2，默认插入 2 周期。

- **TXSR: 退出自刷新到激活延迟**

复位值为 5 周期。

该域定义 SCKE 置高与激活命令间的延迟数，值在 1/2 ~ 15.5 间。

若 TXSR 等于 0，默认插入 1/2 周期。



SDRAMC 自刷新寄存器

寄存器名称 :SDRAMC_SRR

访问类型 : 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	–	–	–	–	–	SRCB

- **SRCB: 自刷新命令位**

0 : 无效。

1:SDRAM 控制器向 SDRAM 器件发出一个自刷新命令 ,SDCK 时钟不动 ,SDCKE 信号置低。当重新访问时 SDRAM 器件离开自刷新模式。

SDRAMC 低功耗寄存器

寄存器名称 :SDRAMC_LPR

访问类型 : 读 / 写

复位值 : 0x0

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	–	–	–	–	–	LPCB

- **LPCB: 低功耗命令位**

0 : 禁止 SDRAM 控制器的低功耗特性 : 未向 SDRAM 器件发出低功耗命令。

1: 每次脉冲访问后 SDRAM 控制器向 SDRAM 器件发出低功耗命令 ,SDCKE 信号置低。当重新访问时 SDRAM 器件离开低功耗模式 , 访问结束再进入低功耗模式。

SDRAMC 中断使能寄存器

寄存器名称 :SDRAMC_IER

访问类型 : 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	–	–	–	–	–	RES

- **RES: 刷新错误状态**

0 : 无效。

1 : 使能刷新错误中断。

SDRAMC 中断禁用寄存器

寄存器名称 :SDRAMC_IDR

访问类型 : 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	–	–	–	–	–	RES

- **RES: 刷新错误状态**

0 : 无效。

1 : 禁用刷新错误中断。



SDRAMC 中断屏蔽寄存器

寄存器名称： SDRAMC_IMR

访问类型： 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	–	–	–	–	–	RES

- **RES: 刷新错误状态**

0：禁用刷新错误中断。

1：使能刷新错误中断。

SDRAMC 中断状态寄存器

寄存器名称： SDRAMC_ISR

访问类型： 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	–	–	–	–	–	RES

- **RES: 刷新错误状态**

0：上次寄存器读后，没有检测到刷新错误。

1：上次寄存器读后，检测到刷新错误。

Burst Flash 控制器 (BFC)

概述

Burst Flash 控制器 (BFC) 向外部 16 位 Burst Flash 器件提供接口，可处理的地址空间为 256M 字节。它支持字节、半字和字访问，最高可访问 32M 字节 Burst Flash 器件。 BFC 还支持数据总线与地址总线复用。 Burst Flash 接口只支持连续脉冲读取。编程脉冲宽不可能为 4 或 8 个字。不管请求地址是否在 256M 字节地址空间内， BFC 都不会产生异常中断信号。

BFC 有两种读脉冲协议，采用何种读脉冲协议由 Burst Flash 器件地址增加是否由信号控制决定 。位于 BFC 接口的 Burst Flash 控制器模式寄存器 (BFC_MR) 用来确定使用编程异步模式还是采用脉冲操作模式。在脉冲模式中，时钟控制地址增长，在每个时钟周期后自动增加地址。但在信号控制地址增长协议中，只有当脉冲地址增强信号激活地址才会增加。当选择地址与数据总线复用模式时，最低的 16 位地址位与数据总线复用。

BFC时钟速度可编程为主机时钟或2或4倍频主机时钟。某些不能连续脉冲读的Burst Flash器件要求页大小处理 (16 字节到 1024 字节)。地址有效后的延迟周期数可达到 16 个周期。输出使能后的延迟周期数在 1 到 3 个时钟周期间。 Burst Flash 控制器可编程延迟或保持当前的访问脉冲。该特性使得其它器件可以共享 BFC 总线而不会影响效率。Burst 模式下，BFC 可不增加延迟就重启一个访问序列。

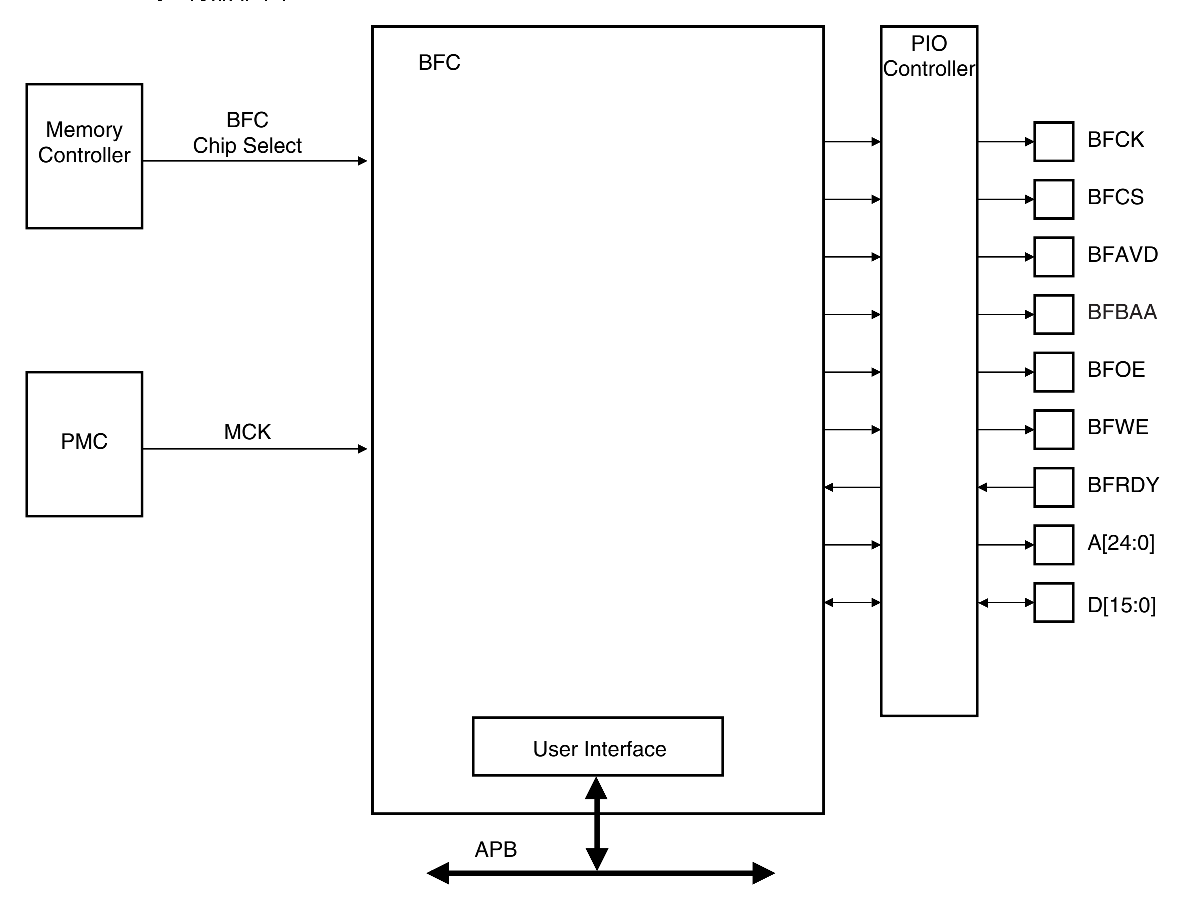
Burst Flash 控制器特性如下：

- 支持多路访问模式
  - 异步或 Burst 模式的字节、半字或字读访问
  - 异步模式半字写访问
- 适应不同器件速度等级
  - 可编程 Burst Flash 时钟速率
  - 可编程数据访问时间
  - 可编程的输出使能后延时
- 适应不同器件访问协议及总线接口
  - 两个脉冲读协议：时钟控制地址增长与信号控制地址增长信号控制地址增长
  - 复用或独立的地址与数据总线
  - 支持连续脉冲与页模式访问



方框图

Figure 96. Burst Flash 控制器框图



I/O 线说明

Table 56. I/O 线说明

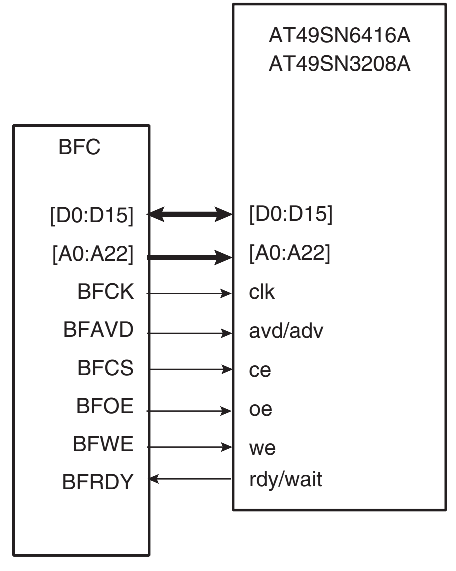
名称	说明	类型	有效电平
BFCK	Burst Flash 时钟	输出	
BFCS	Burst Flash 片选	输出	低
BFAVD	Burst Flash 地址有效	输出	低
BFBAA	Burst Flash 增强地址	输出	低
BFOE	Burst Flash 输出使能	输出	低
BFWE	Burst Flash 写使能	输出	低
BFRDY	Burst Flash 就绪	输入	高
A[24:0]	地址总线	输出	
D[15:0]	数据总线	I/O	

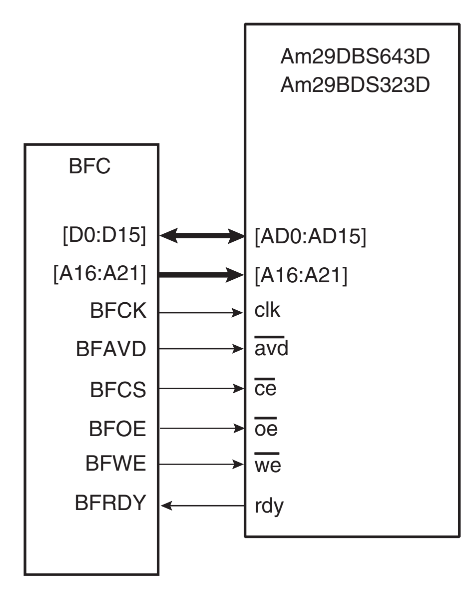
应用实例

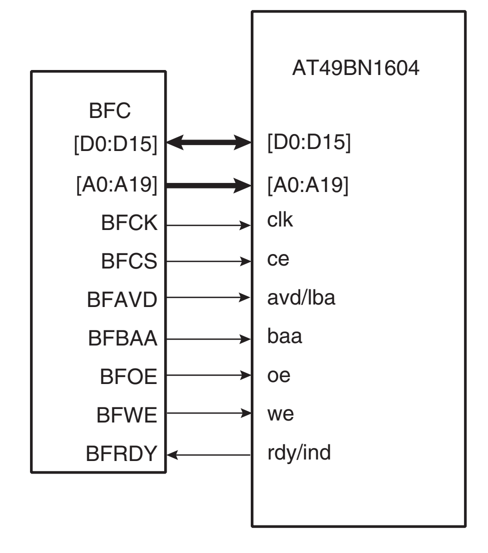
Burst Flash 接口

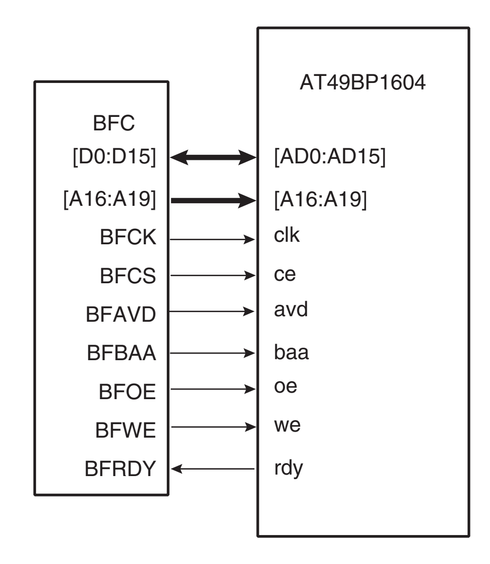
Burst Flash接口向Burst Flash存储器提供控制、地址及数据信号。BFC功能与操作模式见“功能说明” on page 210。 Figure 97 给出 BFC 与某些通用 Burst Flash 存储器的一种连接方式。

Figure 97. Burst Flash 控制器连接示例











相关产品

支持 Burst Flash 器件

Burst Flash 控制器设计优先支持下述 ATMEL Burst Flash 器件：

- AT49SN6416A 和 AT49SN6416AT (64 Mbits x 16)
- AT49SN3208A 和 AT49SN3208AT (32 Mbits x 16)
- AT49BN3208 和 AT49BN3208T (32 Mbits x 16)

I/O 线

Burst Flash控制器接口引脚可与PIO线复用。程序员必须先对PIO控制器编程来分配Burst Flash 控制器引脚的外设功能。若应用程序没有使用 Burst Flash 控制器的 I/O 口线，它们可被 PIO 控制器用作其它用途。

功能说明

Burst Flash 控制器驱动下列信号：

- 地址有效 (BFAVD)，用以锁存地址
- 时钟 (BFCK)，用于脉冲时钟
- 脉冲前沿地址 (BFBAA)，当编程为信号控制前沿脉冲时控制 Burst Flash 存储器地址前沿
- 写使能 (BFWE)，写 Burst Flash 器件
- 输出使能 (BFOE)，使能外部器件数据驱动数据总线

使能后， BFC 驱动地址总线、数据总线及片选 (BFCS) 线。将就绪信号 (BFRDY) 作为输入并作为下一个数据有效性的指示器。

Burst Flash 控制器复位状态

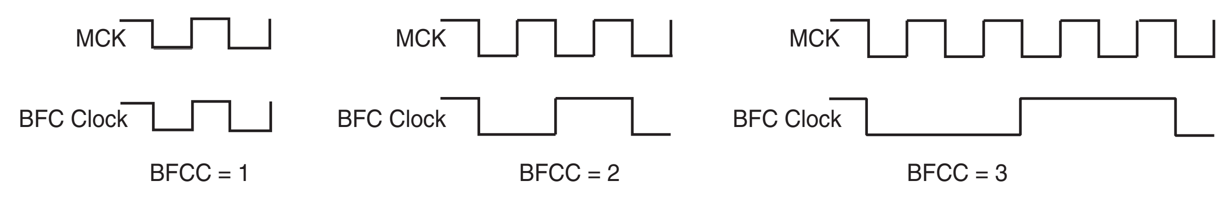
复位后 BFC 被禁用，因此需对 BFCOM 域编程使能，见 See “Burst Flash 控制器模式寄存器 ” on page 218.。此时 Burst Flash 控制器工作在异步模式下。Burst Flash 存储器可在异步模式下进行读、写操作。

Burst Flash 控制器时钟选择

BFC 时钟速率可编程为主机时钟或 2 或 4 倍频主机时钟。脉冲模式与异步模式均需要时钟选择。模式寄存器中的延迟域及所有 burst Flash 控制信号波形与 Burst Flash 时钟 (BFCK) 周期相关。

BFC 时钟速率由 BFCC 域选定，见 See “Burst Flash 控制器模式寄存器 ” on page 218.。

Figure 98. Burst Flash 时钟速率



Burst Flash 控制器异步模式

异步模式下， Burst Flash 控制器时钟关闭。 BFCK 信号拉低。

BFC执行字节(8位)、半字(16位)及字(32位)读访问。在最后一种情况中 ,BFC自动将字读请求转换成两个独立的半字读。这对用户是透明的。

BFC 只执行半字写请求。字节或字的写请求被 BFC 忽略。

任何对地址空间的访问，当 BFAVD 上有一个脉冲信号时地址总线驱动地址（见 Figure 99 on page 211 及 Figure 100 on page 212)。当数据与地址总线复用使能时 ,Burst Flash 地址也可由数据总线驱动 (Figure 99 on page 211)。

- 写访问时，BFWE 信号在 BFCK 时钟周期后出现。
- 读访问时，BFOE 信号在一个周期后出现。读访问中的这个附加周期用来切换 I/O 焊盘方向以避免地址与数据总线复用时 Burst Flash 数据总线冲突。

地址有效延迟 (AVL) 确定脉冲长度的主机时钟周期数。 AVL 域值为地址有效延迟减 1(见 See "Burst Flash 控制器模式寄存器 " on page 218.)。Figure 99 on page 211 与 Figure 100 on page 212 给出读、写访问时 AVL 域定义波形。

- 读访问时，访问在 BFOE 的上升沿结束。
- 读访问时，数据与地址线在 BFWE 上升沿后 0.5 周期后释放。

对 Burst Flash 读访问后， Burst Flash 器件在输出使能延迟 (OEL) 周期中释放数据总线。 OEL 域 (见 See "Burst Flash 控制器模式寄存器 " on page 218.) 在 BFCK 时钟周期中给出 OEL 表示。这防止在 Burst Flash 器件释放数据总线前，其它存储控制器对数据总线的使用。

Figure 99 on page 211 中 (地址与数据总线复用) , 在读与写访问间插入一个空周期 (OEL = 1)。在 BFC 能驱动地址前 Burst Flash 器件必须释放数据总线。如 Figure 100 on page 212 所示，总线未复用时 , 读访问结束后立即启动写访问。同样 , 当写访问后紧跟一个读访问时 OEL 无效。

Figure 99 on page 211 及 Figure 100 on page 212 波形与 Burst Flash 控制器时钟相关。BFCC 域 (见 See "Burst Flash 控制器模式寄存器" on page 218.) 用来测量 burst Flash 速度 , 必须在异步模式下编程。

Figure 99. 地址与数据总线复用时的异步读写访问

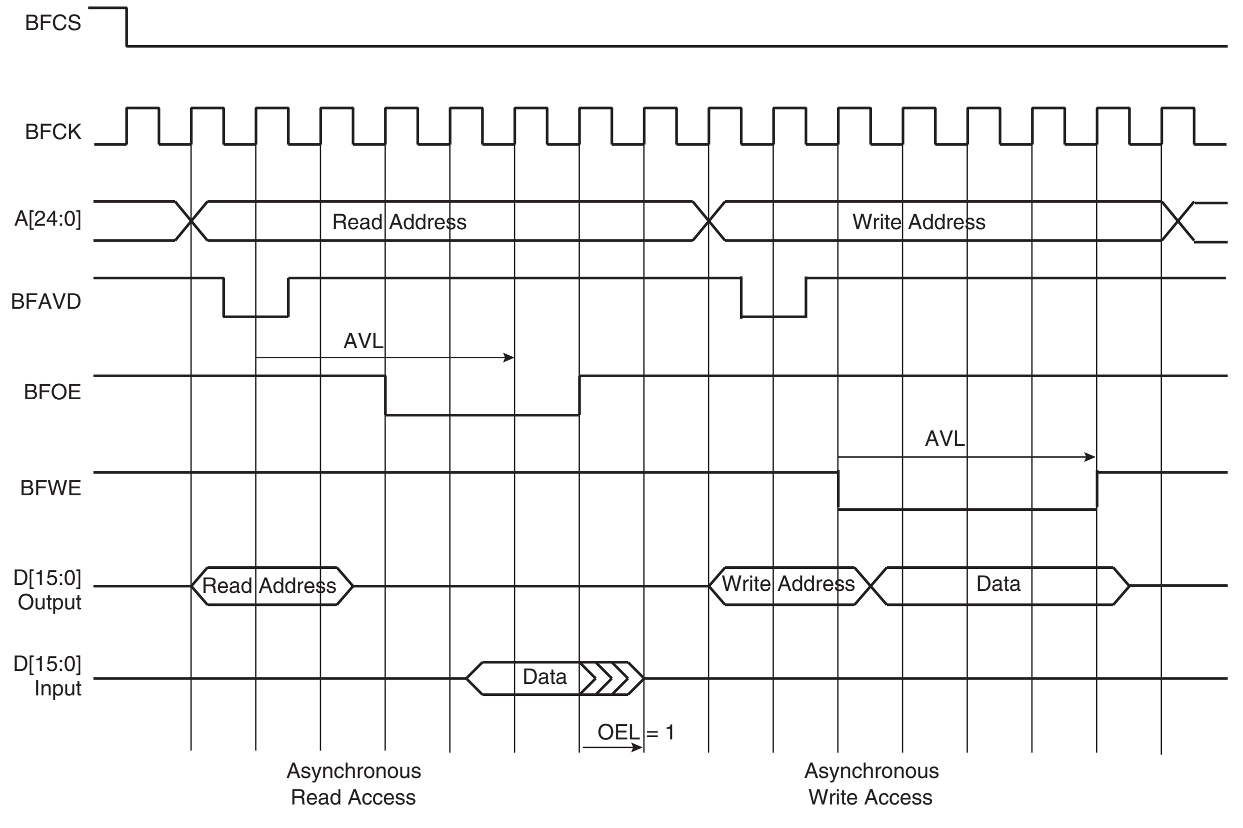
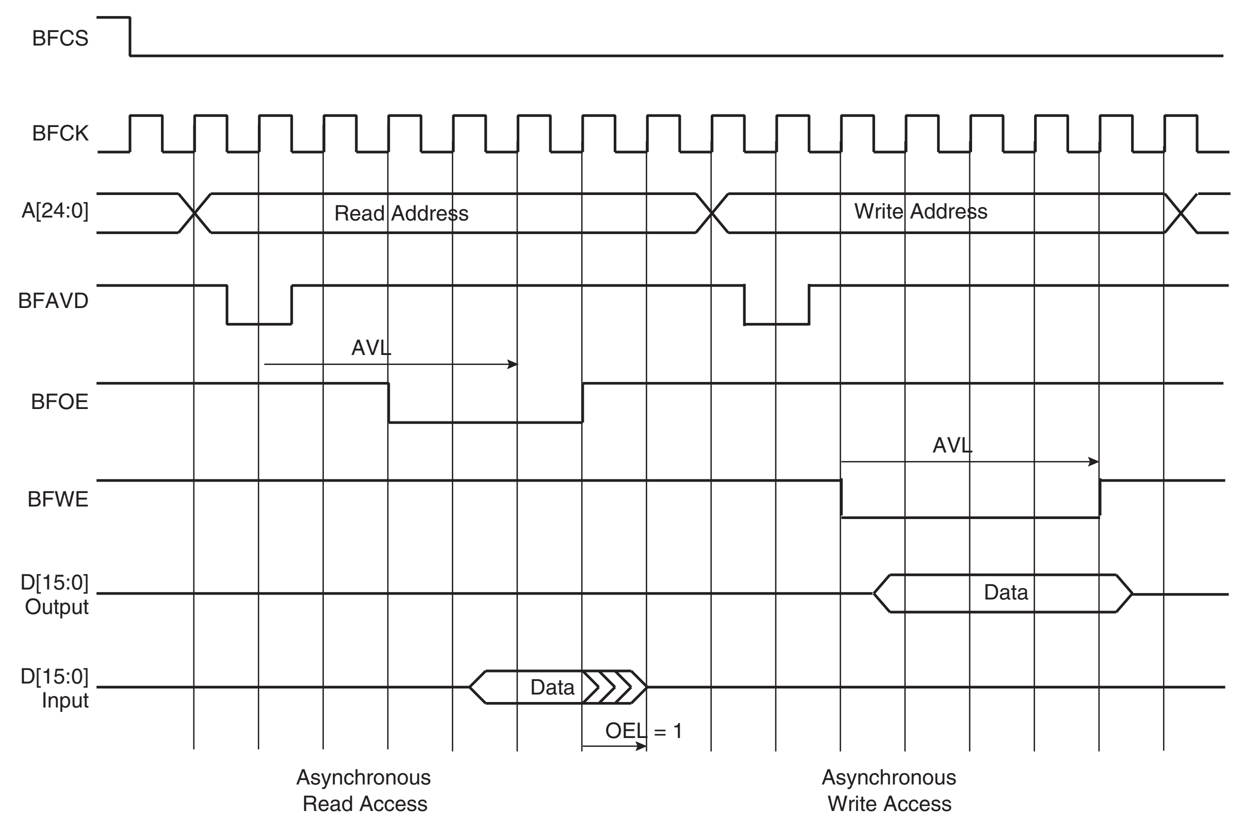




Figure 100. 地址与数据总线未复用时的异步读写访问



Burst Flash 控制器同步模式

在 Burst Flash 控制器操作模式域 (BFCOM) 写入 2(见 "Burst Flash 控制器模式寄存器 " on page 218) 设置BFC工作在脉冲模式下。BFCK引脚驱动BFC时钟。仅处理读访问而忽略写访问。BFC 支持字节、半字及字的读访问

脉冲读协议

BFC 支持两种脉冲读协议 :

- 时钟控制地址增长， burst Flash 内部地址在每个 BFCK 时钟后自动加一。
- 信号控制地址增长， burst Flash 内部地址在 BFBAA 激活后加一。

脉冲模式下的读访问

脉冲模式下读访问请求时，请求地址存于 BFC 中。对于后来的读访问，其地址与前一个进行比较。考虑下面两种情况 :

1. 若后面没有访问，当前脉冲中断， BFC 执行一个地址锁存周期启动一个新脉冲。任何情况下地址都出现在地址总线上，当总线复用使能时也会在数据总线上出现。新地址存于 BFC 中，作为以后访问的参考。
2. 若后面还有访问 ,并在模式寄存器中选择 BFOEH 模式 (见 See "Burst Flash 控制器模式寄存器 " on page 218.)，内部脉冲地址加一。
   – 若信号控制地址增长使能，通过 BFBAA 引脚实现。
   – 若时钟控制地址增长使能，在一个时钟周期内使能时钟。

这些协议见 Figure 101 与 Figure 102 on page 214。地址有效延迟 (AVL+1，见 "Burst Flash 控制器模式寄存器 " on page 218) 给出从 BFAVD 出现后的第一个时钟上升沿到读数据 D1 的下降沿间的时钟数。

Note: 该上升沿还将 D0 锁存在 BFC 中。

Figure 101. 信号控制地址增长模式下脉冲延迟与恢复

BFCS

Internal BFC Selection Signal

BFCK

A[24:0] Address (D0)

BFAVD

AVL

OEL = 2

BFOE

BFWE

Burst Suspend

Burst Resume

BFBAA

D[15:0] Output Address (1)

D[15:0] Input D0 D1 D2 D3 D4 D4 D5 D6

D0 Sampling

D2 Sampling

D5 Sampling

D1 Sampling

D3 Sampling

D4 Sampling

Burst Suspend and Resume (BFOEH = 1)
Signal Control Address Advance (BAAEN = 1)
(1) Only if Multiplexed Address & Data Buses

Address Valid Latency = 4 BFCK cycles (AVL field = 3)
Output Enable Latency (OEL) = 2 BFCK cycles



Figure 102. 时钟控制地址增长模式下脉冲延迟与恢复

Burst Suspend and Resume (BFOEH = 1)
Clock Control Address Advance (BAAEN = 0)
(1) Only if Multiplexed Address & Data Buses

Address Valid Latency = 4 BFCK cycles (AVL = 3)
Output Enable Latency (OEL) = 2 BFCK cycles

脉冲暂停来使能传输

BFC 能暂停脉冲以使能其它内部传输，或使能与其共享地址与数据总线的其它存储控制器。BFOEH 位有两种模式 (Burst Flash 输出使能处理，见 “Burst Flash 控制器模式寄存器 ” on page 218)：

- BFOEH = 1 : 当不选择脉冲且 BFOE 引脚待定时，BFC 暂停脉冲。当 Burst Flash 器件上有连续新的访问请求时，脉冲恢复且 BFOE 重新出现。 OEL 周期后的数据有效。该模式提供最小的访问延迟 (参见 Figure 101 on page 213 及 Figure 102)。
- BFOEH = 0 : 当不选择脉冲且 BFOE 引脚待定时，BFC 暂停脉冲。当 Burst Flash 器件上有新的访问请求时，不论是否连续，初始化一个新的脉冲且在模式寄存器 AVL 延迟域中定义的下一个数据有效。该模式用于无效的 BFOE 信号引起的脉冲不可逆中断的 Burst Flash 器件。 Figure 103 on page 215 给出 BFC 访问请求及由于 BFC (延迟) 未选引起的 BFOE 信号无效。当再次请求 BFC，即使请求地址同前一个请求地址连续，重新启动一个新脉冲。

Figure 103. 无脉冲使能处理的 Burst Flash 控制器

Address Valid Latency = 4 BFCK cycles
Output Enable Latency (OEL) = 1 BFCK cycle

(1) Only if Multiplexed Address & Data Busses
(2) Master Clock Mode (BFCC =1)

No Burst Output Enable Handling (BFOEH = 0)
Signal Control Advance Address (BAAEN = 1)

连续脉冲读 BFC可执行连续脉冲读。它可编程的页面大小从16字节到1024字节。这可通过设置PAGES域值实现，见 “Burst Flash 控制器模式寄存器 ” on page 218。

页模式 页模式下，当请求地址到达页边界时，BFC 停止当前脉冲并启动一个新的脉冲。Figure 104 on page 216 给出一个 16 字节的页。数据 D0 到 D10 属于两个独立的页，通过两个脉冲访问进行访问。该模式是针对不能处理连续脉冲读的 Burst Flash 器件 (此时对地址 D0 的连续脉冲访问将引起 Burst Flash 内部地址翻转回 D0 地址)。对 PAGES 域写入零将禁用页模式，见 “Burst Flash 控制器模式寄存器 ” on page 218。



Figure 104. 页模式下的脉冲读

就绪使能模式

就绪使能模式下 (“Burst Flash 控制器模式寄存器 ” on page 218 中的位 RDYEN)，BFC 使用来自 burst Flash 的就绪信号 (BFRDY) 作为下一个数据有效的指示器 。数据有效前 BFRDY 信号必须插入一个 BFCK 周期。如 Figure 105 on page 217 所示 ,BFRDY 信号指出处于边沿 (A) 预期的 D4 数据在下一个 BFCK 上升沿处无效。BFRDY 信号将保持低电平到上升沿 (B)。 D4 在边沿 (C) 处采样。

当 RDYEN 模式禁用 (RDYEN = 0)，忽略 BFC 输入接口的 BFRDY 信号。该模式用于不处理 BFRDY 信号的 Burst Flash 器件。

Figure 105. 使用 BFRDY 信号的脉冲读

(A) (B) (C)

BFCS

BFCK

A[24:0] Address (D0)

BFAVD

AVL

BFOE

D[15:0] Input D0 D1 D2 D3 D4 D5 D6 D7

BFBAA

BFRDY

Sampling D0 D1 D2 D3 D4 D5 D6 D7

Burst Read
Signal Control Advance Address (BAAEN = 1)

Address Valid Latency = 4 BFCK cycles (AVL field = 3)
Output Enable Latency (OEL) = 1 BFCK cycle



Burst Flash 控制器 (BFC) 用户接口

Burst Flash 控制器模式寄存器

寄存器名称 : BFC_MR

访问类型: 读 / 写

复位值 : 0x0

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	RDYEN	MUXEN	BFOEH	BAAEN

15	14	13	12	11	10	9	8
–	–	OEL			PAGES		

7	6	5	4	3	2	1	0
AVL				BFCC		BFCOM	

- **BFCOM: Burst Flash 控制器操作模式**

BFCOM		BFC 操作模式
0	0	禁用 .
0	1	异步
1	0	脉冲读
1	1	保留

- **BFCC: Burst Flash 控制器时钟**

BFCC		BFC 时钟
0	0	保留
0	1	主机时钟
1	0	主机时钟 2 倍频
1	1	主机时钟 4 倍频

- **AVL: 地址有效延迟**

地址有效延迟定义从 BFAVD 插入后的第一个 BFCK 上升沿到采样读数据的 BFCK 上升沿的时钟数。延迟时间等于 AVL + 1。

- **PAGES: 页大小**

该域定义处理的页大小及页大小

页			页大小
0	0	0	无页处理。采样就绪信号 (BFRDY) 来检测下一个数据是否有效
0	0	1	16 字节页大小
0	1	0	32 字节页大小
0	1	1	64 字节页大小
1	0	0	128 字节页大小
1	0	1	256 字节页大小
1	1	0	512 字节页大小
1	1	1	1024 字节页大小

- **OEL: 输出使能延迟**

该域定义 BFOE 输出使能信号后每次电平改变插入的空闲周期数。 OEL 范围为从 1 到 3。

- **BAAEN: 脉冲地址增长使能**

0：脉冲时钟使能增加脉冲地址或禁止以保持地址不变。

1：脉冲时钟连续且由 BFBAA 引脚控制脉冲地址增长。

- **BFOEH: Burst Flash 输出使能处理**

0：脉冲模式中无脉冲恢复。当 BFC 未选，引起脉冲的不可逆中断。为下次访问初始化新脉冲。

1：脉冲恢复。当 BFC 未选，脉冲延迟。若新访问与上次访问连，脉冲将恢复。

- **MUXEN: 复用总线使能**

0：地址与数据总线独立操作。

1：地址与数据总线复用。实际上当 BFAVD 信号出现后地址会同时出现在数据总线与地址总线上。

- **RDYEN: 就绪使能模式**

0：BFC 输入接口的 BFRDY 输入信号忽略。

1：BFRDY 输入信号作为下一个周期数据有效性的检测器。



外设数据控制器 (PDC)

概述

外设数据控制器 (PDC) 在诸如 UART、 USART、 SSC、 SPI、 MC 等片上串行外设与片内、片外存储器间传输数据。使用外设数据控制器能避免处理器干涉并删除处理器中断处理开销。这显著的降低了数据传输所需的时钟周期数，因此也改善了微控制器的性能，使其更有效。

PDC 通道是成对构建的，每对对应一个指定的外设。通道中一个负责接收、另一个负责发送。

PDC 用户接口集成在每个外设存储空间中。它包括：

- 1 个 32 位存储器指针寄存器
- 1 个 16 位传输计数寄存器
- 1 个 32 位寄存器，用作下个存储器指针
- 1 个 16 位寄存器，用作下个传输计数

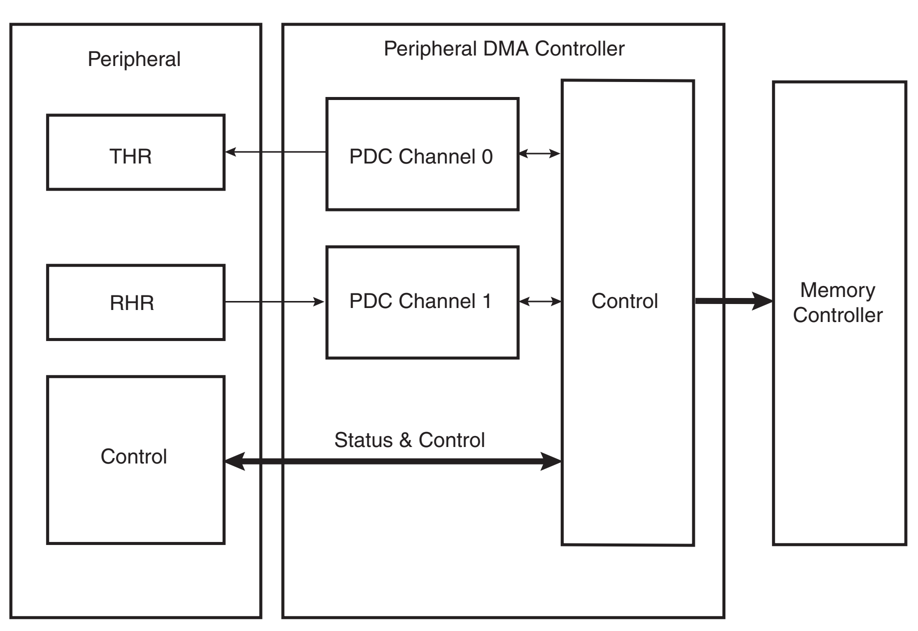
外设使用发送与接收信号触发 PDC 传输。当传输编程数据时，相应的外设产生传输中断结束。

PDC 特性如下：

- 与诸如 DBGU、 USART、 SSC、 SPI 及 MCI 外设间收发
- 最高支持 20 个通道 (取决于产品本身)
- 存储器到外设间的传输须一个主机时钟周期
- 外设到存储器间的传输须两个主机时钟周期

方框图

Figure 106. 方框图



功能说明

配置

PDC 通道用户接口使能用户配置和控制各个通道的数据传输。PDC 通道的用户接口集成在与其相关的外设用户接口上 (偏移 0x100)。

每个外设包含四个 32 位指针寄存器 (RPR、 RNPR、 TPR 及 TNPR) 与四个 16 位计数寄存器 (RCR、 RNCR、 TCR 及 TNCR)。





缓冲器的大小 (传输数) 在内部 16 位传输计数寄存器中配置，可在任意时刻读取每个通道的待传输数据的大小。

通过定义存储器访问的首地址位置，在 32 位存储器指针配置存储器基地址。可在任意时刻读取下一次传输的存储空间位置和待传输的数目。PDC 有专门的状态寄存器指示各通道传输是否使能。各通道状态位于外设状态寄存器中。通过设置 PDC 传输控制寄存器的 TXTEN/TXTDIS 与 RXTEN/RXTDIS 可使能或禁用传输。这些控制位使能安全读取指针及计数寄存器，在读取的过程中改变它们而不会有危险。

PDC 向外设发送的状态标志在其状态寄存器中可见 (ENDRX、ENDTX、RXBUFF 及 TXBUFE)。

当 PERIPH_RCR 寄存器为零时设置 ENDRX 标志。

当 PERIPH_RCR 与 PERIPH_RNCR 为零时设置 RXBUFF 标志。

当 PERIPH_TCR 寄存器为零时设置 ENDTX 标志。

当 PERIPH_TCR 与 PERIPH_TNCR 为零时设置 TXBUFE 标志。

这些状态标志在外设状态寄存器中。

存储器指针

每个外设通过接收器数据通道及接收器数据通道与 PDC 连接。每个通道有一个内部的 32 位存储器指针。每个存储器指针可指向存储空间的任意位置 (片上存储器或外部总线接口存储器)。

根据传输类型的不同 (字节、半字或字)，外设传输的存储器指针以 1、 2 或 4 增加。

若 PDC 工作时对存储器指针重编程，传输地址改变，在新地址执行 PDC 传输。

传输计数器

每个通道有一个内部 16 位传输计数器用来计算已传输块的大小。每次数据传输完成后计数器减一。当计数器到零时，传输完成， PDC 停止传输数据。

若下一个计数寄存器等于零， PDC 禁用触发并激活相关的外设结束标志。

若 PDC 工作时对存储器重编程，传输数目更新， PDC 由新值开始计数。

编程下一个计数器 / 指针寄存器链接到缓冲器。计数器在每次数据传输结束后减一，但当传输计数器达到零时，下一个计数器 / 指针载入计数器 / 指针寄存器中以重新使能触发器。

各个通道中，有两个状态位来标识当前缓冲结束 (ENDRX， ENTX) 及当前缓冲与下次缓冲结束 (RXBUFF， TXBUFE)。这些位直接映射到外设状态寄存器中并可触发到 AIC 的中断请求。

当写入一个计数器寄存器 (计数器或下次计数器) 时，外设结束标志自动清零。

注意：当下次计数寄存器值载入到计数寄存器中后，其值设为 0。

数据传输

使用发送 (TXRDY) 与接收 (RXRDY) 信号触发外设 PDC 传输。

当外设接收到一个外部字符时，它向 PDC 发送一个接收就绪信号， PDC 再向系统总线请求访问。当访问得到许可， PDC 开始读取外设接收保持寄存器 (RHR) 并触发存储器写操作。

每次传输后，相关 PDC 存储器指针递增，而待传输数目递减。当达到存储器块大小时，向外设发送一个信号并停止传输。

对发送传输，接下来的处理步骤相反。

PDC 传输请求优先级

外设数据控制器通过固定各个产品优先级来处理来自通道的传递请求。优先级在产品数据手册上有定义。

若在同一外设上同时出现相同类型的请求 (接收器或发送器) 优先级由外设编号决定。

若传输请求不是同时出现，则按其出现的先后顺序进行处理。接收器请求先行处理，然后处理发送器请求。

外设数据控制器 (PDC) 用户接口

Table 57. 寄存器映射

偏移	寄存器	寄存器名称	访问	复位
0x100	PDC 接收指针寄存器	PERIPH[1]_RPR	读 / 写	0x0
0x104	PDC 接收计数寄存器	PERIPH_RCR	读 / 写	0x0
0x108	PDC 传输指针寄存器	PERIPH_TPR	读 / 写	0x0
0x10C	PDC 传输计数寄存器	PERIPH_TCR	读 / 写	0x0
0x110	PDC 接收下次指针寄存器	PERIPH_RNPR	读 / 写	0x0
0x114	PDC 接收下次计数寄存器	PERIPH_RNCR	读 / 写	0x0
0x118	PDC 发送下次指针寄存器	PERIPH_TNPR	读 / 写	0x0
0x11C	PDC 发送下次计数寄存器	PERIPH_TNCR	读 / 写	0x0
0x120	PDC 传输控制寄存器	PERIPH_PTCR	只写	-
0x114	PDC 传输状态寄存器	PERIPH_PTSR	只读	0x0

Note: 1. PERIPH : 10 个寄存器可映射到相同映射偏移地址的外设存储空间中，用户可根据功能与外设需求 (DBGU、USART、SSC、SPI、 MCI 等) 进行定义。

PDC 接收指针寄存器

寄存器名称：PERIPH_RPR

访问类型 :读 / 写

31	30	29	28	27	26	25	24
RXPTR							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
RXPTR							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RXPTR							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RXPTR							

- **RXPTR: 接收指针地址**

下次接收传输的地址。

PDC 接收计数寄存器

寄存器名称： PERIPH_RCR

访问类型：读 / 写

31	30	29	28	27	26	25	24
--							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
--							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RXCTR							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RXCTR							

- **RXCTR: 接收计数器值**

执行接收传输的数目。

PDC 发送指针寄存器

寄存器名称：PERIPH_TPR

访问类型：读 / 写

31	30	29	28	27	26	25	24
TXPTR							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
TXPTR							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
TXPTR							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TXPTR							

- **TXPTR: 发送指针地址**

发送缓冲器地址。

PDC 发送计数寄存器

寄存器名称： PERIPH_TCR

访问类型：读 / 写

31	30	29	28	27	26	25	24
--							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
--							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
TXCTR							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TXCTR							

- **TXCTR: 发送计数器值**

·TXCTR 是执行发送传输的大小。为零时，外设数据传输停止。

PDC 接收下次指针寄存器

寄存器名称：PERIPH_RNPR

访问类型 :读 / 写

31	30	29	28	27	26	25	24
RXNPTR							
23	22	21	20	19	18	17	16
RXNPTR							
15	14	13	12	11	10	9	8
RXNPTR							
7	6	5	4	3	2	1	0
RXNPTR							

- **RXNPTR: 接收下次指针地址**

RXNPTR 是在当前缓冲器满时，接收数据要填充的下一个缓冲器地址。

PDC 接收下次计数寄存器

寄存器名称：PERIPH_RNCR

访问类型 :读 / 写

31	30	29	28	27	26	25	24
--							
23	22	21	20	19	18	17	16
--							
15	14	13	12	11	10	9	8
RXNCR							
7	6	5	4	3	2	1	0
RXNCR							

- **RXNCR: 接收下次计数器值**

·RXNCR 为下次要接收的缓冲器大小

PDC 发送下次指针寄存器

寄存器名称：PERIPH_TNPR

访问类型 :读 / 写

31	30	29	28	27	26	25	24
TXNPTR							
23	22	21	20	19	18	17	16
TXNPTR							
15	14	13	12	11	10	9	8
TXNPTR							
7	6	5	4	3	2	1	0
TXNPTR							

- **TXNPTR: 发送下次指针地址**

TXNPTR 是在当前缓冲器空时，下次要发送的缓冲器地址。



PDC 发送下次计数寄存器

寄存器名称： PERIPH_TNCR

访问类型：读 / 写

31	30	29	28	27	26	25	24
--							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
--							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
TXNCR							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TXNCR							

- **TXNCR: 发送下次计数器值**

·TXNCR 为下个缓冲器要发送的大小。

PDC 传输控制寄存器

寄存器名称： PERIPH_PTCR

访问类型：只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	TXTDIS	TXTEN
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	–	–	–	–	RXTDIS	RXTEN

- **·RXTEN: 接收传输使能**

0 = 无效。

1 = 使能接收器 PDC 传输请求，若 RXTDIS 未设置。

- **·RXTDIS: 接收器传输禁用**

0 = 无效。

1 = 禁用接收器 PDC 传输请求。

- **·TXTEN: 发送器传输使能**

0 = 无效。

1 = 使能发送器 PDC 传输请求。

- **·TXTDIS: 发送器传输禁用**

0 = 无效。

1 = 禁用发送器 PDC 传输请求。

PDC 传输状态寄存器

寄存器名称 : PERIPH_PTSR

访问类型 : 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	TXTEN
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	–	–	–	–	–	RXTEN

- **·RXTEN: 接收器传输使能**

0 = 接收器 PDC 传输请求禁用。

1 = 接收器 PDC 传输请求使能。

- **·TXTEN: 发送器传输使能**

0 = 发送器 PDC 传输请求禁用。

1 = 发送器 PDC 传输请求使能。



高级中断控制器 (AIC)

概述

高级中断控制器 (AIC) 是一个有 8 个优先级，独立可屏蔽的向量中断控制器，最高可处理 32 个中断源。它的设计充分降低了软件与实时开销内部与外部中断处理。

AIC驱动ARM处理器nFIQ (快速中断请求)与nIRQ (标准中断请求)输入。AIC输入可以是内部外设中断或来自器件引脚的外部中断。

8级优先级控制器允许用户对每个中断源优先级进行定义，允许高优先级中断打断低优先级中断先行处理。

内部中断源可编程为电平敏感或边沿触发。外部中断源可编程为正沿或负沿触发或高电平或低电平敏感。

快速强制特性可改变内部或外部中断源提供一个快速中断。

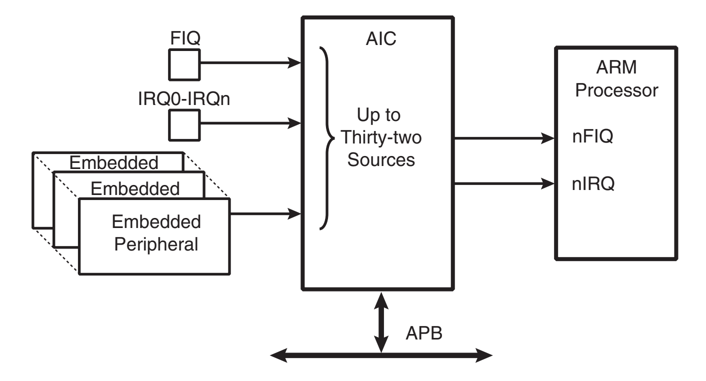
AIC 特性如下：

- 控制 ARM® 处理器中断线 (nIRQ 与 nFIQ)
- 32 个独立可屏蔽向量中断源
  – 源 0 为快速中断输入 (FIQ)
  – 源 1 为系统外设 (ST、 RTC、 PMC、 DBGU…)
  – 源 2 到源 31 控制多达 30 个内置外设中断或外部中断
  – 可编程边沿触发或电平敏感内部源
  – 可编程正 / 负边沿触发或高 / 低电平敏感外部源
- 8 级优先级控制器
  – 驱动处理器正常中断
  – 处理中断源 1 到 31 的优先级
  – 高优先级中断可在低优先级中断服务期间先行服务
- 向量
  – 优化中断服务程序分支与执行
  – 每个中断源有一个 32 位向量寄存器
  – 中断向量寄存器读相应当前中断向量
- 保护模式
  – 当保护模式使能时通过防止自动操作而方便调试
- øÏÀŸ«ø÷ý
  – 允许重定向处理器快速中断的普通中断源
- 通用中断屏蔽
  – 提供不触发中断的处理器同步事件



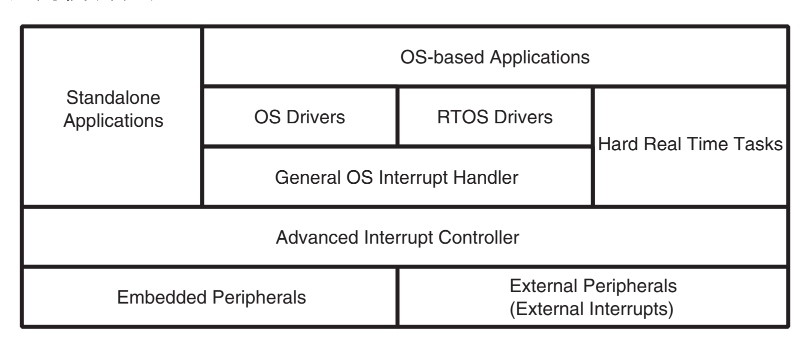
方框图

Figure 107. 方框图



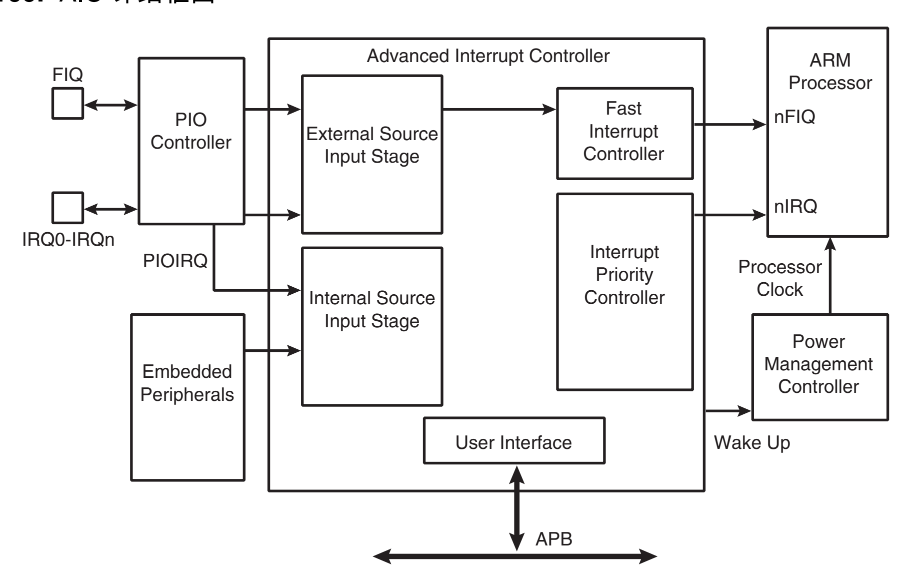
应用方框图

Figure 108. 应用模块说明



AIC 详细框图

Figure 109. AIC 详细框图



I/O 线说明

Table 58. I/O 线说明

引脚名称	引脚说明	类型
FIQ	快速中断	输入
IRQ0 - IRQn	中断 0 - 中断 n	输入

Product Dependencies

I/O 线

中断信号 FIQ 与 IRQ0 到 IRQn 一般通过 PIO 控制器复用。取决于所用产品的 PIO 控制器特性，引脚必须根据其分配的中断功能进行编程。当 PIO 控制器输入路径透明时不可用。

电源管理

高级中断控制器是连续时钟驱动的。电源管理控制器对高级中断控制器无效。

高级中断控制器输出出现 (nIRQ 或 nFIQ) 用以唤醒处于空闲模式的 ARM 处理器。通用中断屏蔽特性使能 AIC 来唤醒处理器，不必插入中断，因此在处理器上提供事件同步。

中断源

中断源 0 定义为 FIQ。若产品没有 FIQ 引脚，中断源 0 不能用。

中断源 1 定义为系统中断。它是诸如系统定时器、实时时钟、电源管理控制器及存储控制器等系统外设中断线或的结果。当系统中断出现，服务程序先辨别中断的起因。这可通过读上述系统外设的状态寄存器来实现。

中断源 2 到 31 可与内置的用户外设或外部中断线连接。外部中断线可直接连接或通过 PIO 控制器连接。

PIO 控制器在中断处理中可看作用户外设。因此， PIO 控制器中断线与中断源 2 到 31 连接。

定义在产品上的外设标识符与中断源号码对应 (即位号控制外设时钟)。因此，为简化功能操作及用户接口说明，中断源命名为 FIQ、 SYS 及 PID2 到 PID31。





功能说明

中断源控制

中断源模式

高级中断控制器对每个中断源独立编程。相应的 AIC_SMR(源模式寄存器)SRCTYPE 域中选择每个源的中断条件。

连接在内置外设的输出中断的内部中断源可编程为电平敏感模式或边沿触发模式。内部中断激活电平对用户不重要。

外部中断源可编程为高电平敏感或低电平敏感模式；或正沿触发模式或负沿触发模式。

中断源使能

包括源 0 中的 FIQ 在内的所有中断源均可通过命令寄存器，AIC_IECR (中断使能命令寄存器) 及 AIC_IDCR (中断禁用命令寄存器) , 使能或禁用。该套寄存器通过一条指令控制使能或禁用。可在 AIC_IMR 寄存器中读取中断屏蔽。禁用中断不会影响其它中断服务。

中断清除与设置

所有编程为边沿触发的中断源 (包括源 0 中的 FIQ) 可通过写 AIC_ISCR 与 AIC_ICCR 寄存器来设置与清零。在电平敏感模式下，清除或设置中断源编程无效。

因为在边沿触发模式下对源编程时 , 软件必须重新初始化记忆电路 , 因此清除操作不必太认真。但设置操作对于自动测试或软件调试目的有效。它还可用在执行软件中断的 AIC 执行。

当读 AIC_IVR(中断向量寄存器) 时， AIC 自动清除当前中断。只有当该操作影响当前中断时，AIC 才会检测该中断源 (见 See "优先级控制器" on page 235.)。自动清除降低中断服务程序入口代码读 AIC_IVR 的请求操作。注意，若中断源在快速定向特性使能且仅认为是 FIQ 源时，禁用自动中断清除 (详见 See " 快速强制 " on page 238.)。

当读 AIC_FVR 时执行自动清除中断源 0。

中断状态

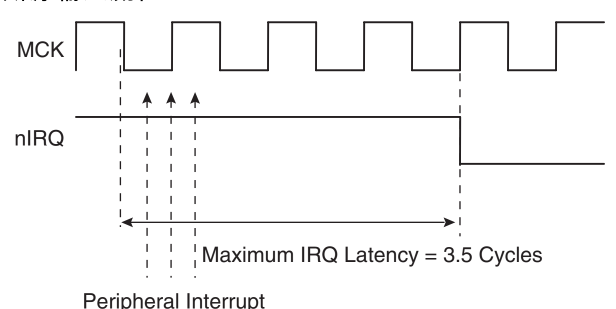
对于每个中断，AIC_IPR(中断挂起寄存器) 与 AIC_IMR (中断屏蔽寄存器) 的屏蔽引起 AIC 操作。 AIC_IPR 使能有效中断源，不管是否被屏蔽。

AIC_ISR寄存器读当前中断数(见"优先级控制器" on page 235) , 寄存器AIC_CISR给出处理器上信号 nIRQ 与 nFIQ 驱动映射。

每个状态可用于优化系统中断处理。

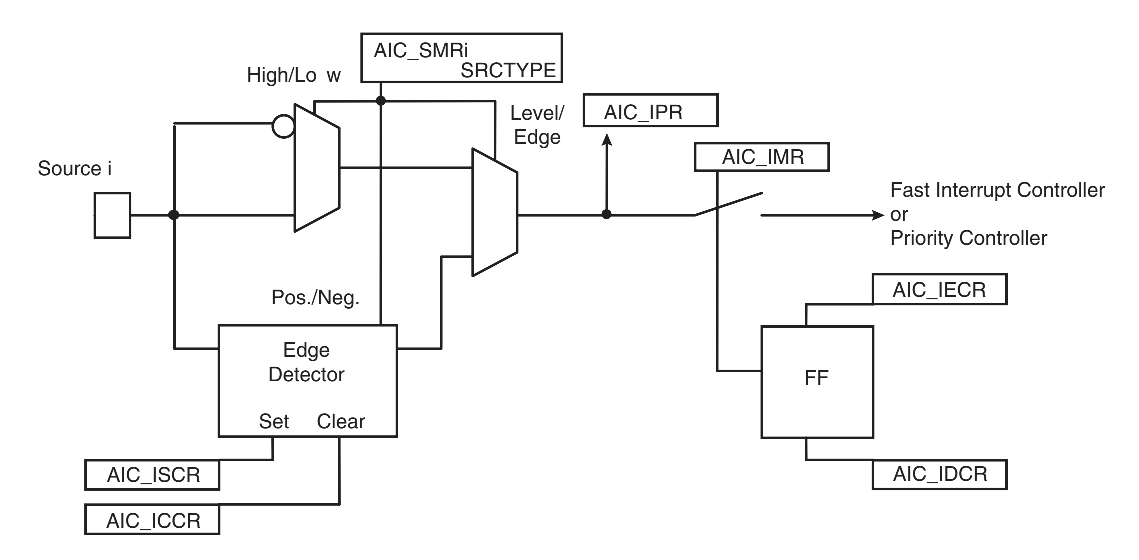
内部中断源输入流程

Figure 110. 内部中断源输入流程



外部中断源输入流程

Figure 111. 外部中断源输入流程



中断延迟

全局中断延迟由以下几个参数决定：

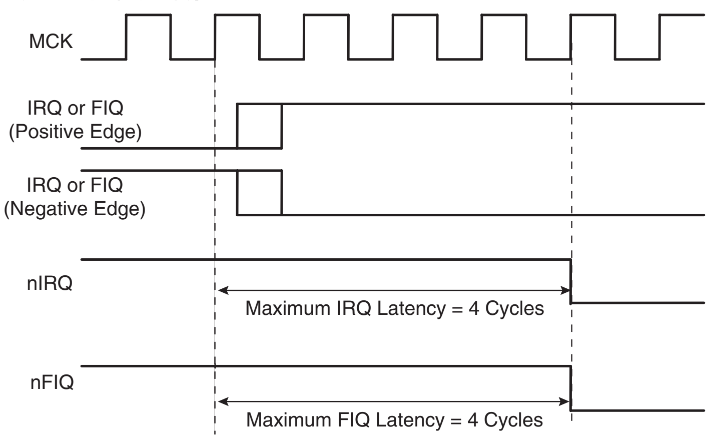
- 软件屏蔽中断时间。
- 处理器级或 AIC 级占用时间。
- 当中断出现时处理中的指令执行时间。
- 高优先级中断处理及硬件信号重同步。

该节仅标注了硬件重同步。给出了有效中断（边沿或电平）引起的外部中断或内部中断源出现事件到处理器 nIRQ 或 nFIQ 出现间延迟时间的详细说明。重同步时间取决于中断源编程及中断源类型（内部或外部）。对于标准中断，给出的重同步时间假设没有处理更高优先级的中断。

PIO 控制器复用对于外部中断源的中断延迟没有影响。

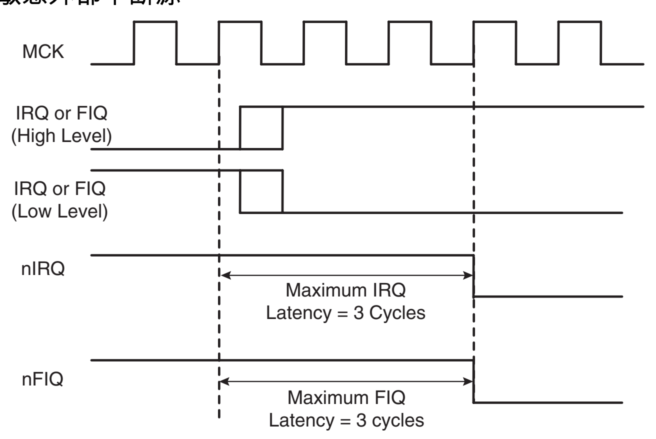
边沿触发外部中断源

Figure 112. 边沿触发外部中断源



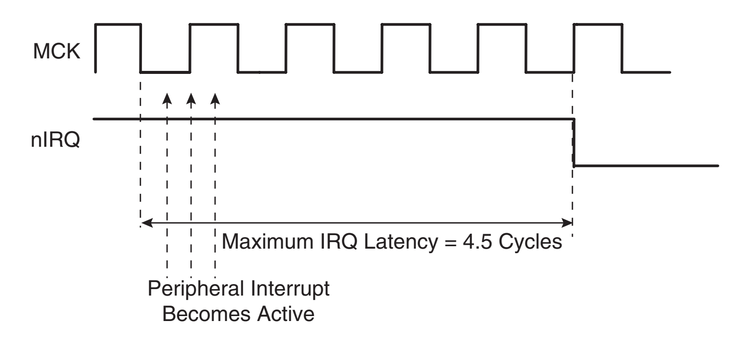
电平敏感外部中断源

Figure 113. 电平敏感外部中断源



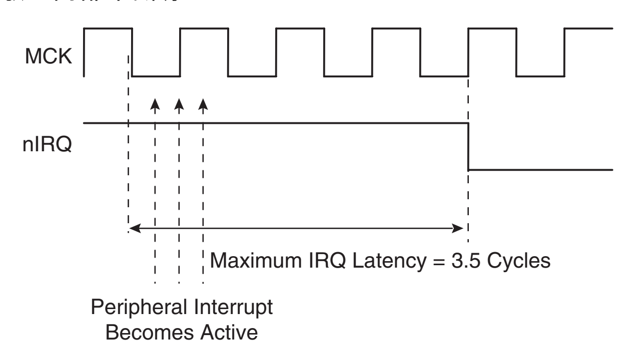
边沿触发内部中断源

Figure 114. 边沿触发内部中断源



电平敏感内部中断源

Figure 115. 电平敏感内部中断源



普通中断

优先级控制器

一个 8 级优先级控制器驱动处理器 nIRQ 线，由出现在中断源 1 到 31 的中断条件决定 (除编程为快速强制的)。

每个中断源通过对相应的 AIC_SMR(源模式寄存器)PRIOR 域写定义其优先级。等级 7 优先级最高，等级 0 优先级最低。

当如 AIC_SVR (源向量寄存器) 中的 SRCTYPE 域定义的中断条件出现时， nIRQ 线出现。由于 nIRQ 出现可能有其它中断源的中断条件出现 , 优先级控制器决定读 AIC_IVR 时的当前中断。对 AIC_IVR 读是中断处理的入口点，允许 AIC 通过软件确定中断。

当前优先级定义为当前中断优先级。

若几个优先级相等的挂起中断源在读 AIC_IVR 后使能，中断源序号最低的中断先行服务。

只有当一个更高优先级的中断源中断条件出现时 , nIRQ 线出现。若在中断执行时出现中断条件 (或挂起)，它将延迟到检测到 AIC 结束当前服务。对 AIC_EOICR 写入将退出中断处理。

中断嵌套

优先级控制器使用中断嵌套以使得在较低优先级中断服务期间可处理高优先级中断服务。这就需要较低优先级中断服务程序可在处理器级重新使能中断处理。

当处理中断服务期间出现一个中断优先级更高的中断， nIRQ 线重新出现。若中断在内核级使能，打断当前中断执行，新的中断服务应读取 AIC_IVR。此时，当前中断序号及其优先级推入内置硬件堆栈，这样它们得以保存并在高优先级中断服务结束并对 AIC_EOICR 写入后重新加载。

AIC 有 8 级硬件堆栈以便支持有 8 级优先级的中断嵌套。



中断向量

对应于每个中断源的中断处理地址可以保存在寄存器AIC_SVR1 到AIC_SVR31 (源向量寄存器 1 到31) 中。当处理器读AIC_IVR (中断向量寄存器) ,返回当前中断相关的写入AIC_SVR中值。

该特性提供了一种单指令处理当前中断的分支方法，由于 AIC_IVR 映射在绝对地址 0xFFFF F100 处，因而 ARM 中断向量通过下面指令访问地址 0x0000 0018：

```
LDR   PC,[PC,# -&F20]
```

当处理器执行该指令时，将 AIC_IVR 读出值载入其程序计数器，然后对正确中断程序进行分支处理。

当应用基于操作系统时 (不论是否实时) 不使用该特性。通常操作系统对于所有中断只有一个入口点且其首先执行的是辨别中断源。

但是，强烈推荐 AT91 产品操作系统，其端口支持中断向量。通过定义所有中断源的 AIC_SVR 在其中断句柄地址处由操作系统处理来实现。此时，中断向量允许在特定的快速句柄执行关键中断，而不是操作系统的通用中断句柄。这使得对硬件实时任务支持更加有效，且在操作系统中独立运行一个应用程序。

中断处理

本节简单给出使用 AIC 时的快速中断处理序列。假设程序员已了解 ARM 处理器架构，特别是处理器中断模式及相关状态位。

假设如下：

1. 高级中断控制器已编程， AIC_SVR 寄存器载入相应的中断服务程序地址且中断使能。
2. ARM 中断异常向量指令地址须与向量一同作用。

```
LDR PC, [PC, # -&F20]
```

当 nIRQ 出现，若 CPSR 中位 “I” 为 0，序列如下：

1. CPSR 存于 SPSR_irq ,程序计数器当前值载入中断链接寄存器 (R14_irq) 且程序计数器 (R15) 中载入 0x18。在由地址 0x1C 取指周期中， ARM 内核调整 R14_irq，以 4 递减。
2. ARM 内核进入中断模式 。
3. 当加载地址为0x18的指令执行时 ,程序计数器载入由AIC_IVR读出的值。读AIC_IVR有以下效果：
   – 将当前中断挂起并使能最高优先级中断。当前级为当前中断优先级。
   – 将处理器上的 nIRQ 线失效。即使向量未用，必须对 AIC_IVR 读以使 nIRQ 失效。
   – 若编程为边沿触发则自动清除中断。
   – 将当前等级及当前中断序号推入堆栈。
   – 返回相应于当前中断写入 AIC_SVR 的值。
4. 先前的步骤已使跳转到相应的中断服务程序。必须先保存链接寄存器 (R14_irq) 及 SPSR_IRQ。若在中断后直接存于程序计数器，链接寄存器值要以 4 递减。例如，使用指令 `SUB PC, LR, #4` 。
5. 后来的中断可通过清除CPSR中的“I”位来不加以屏蔽 ,使得内核又可重新考虑nIRQ。这在出现了一个高于当前中断优先级的中断时产生。
6. 中断处理程序可按要求执行，开始时保存寄存器，结束时恢复它们的值。在此过程中，高于当前中断优先级的中断将使序列返回步骤 1。

Note: 若中断编程为电平敏感，中断源必须在此过程中清除。

7. 必须将 CPSR 中 “I” 位置位以使在退出前屏蔽中断，保证中断严格按照步骤完成。
8. 须写中断结束命令寄存器 (AIC_EOICR) ,以指示 AIC 当前中断已完成。这将使得当前优先级由堆栈中弹出，恢复先前优先级。若有其它中断挂起，等于或低于原先优先级却高于当前新的优先级， nIRQ 线出现出现，但由于 “I” 由内核设置，因此不会立即启动中断序列。恢复 SPSR_irq，最后存于链接寄存器中的值直接写入 PC。这对下面情况有

用：中断返回到中断执行前的状态，将存于 SPSR 中的值载入 CPSR，根据保存在 SPSR_irq 中的状态决定是否屏蔽中断。

Note: SPSR 中的 “I” 位非常重要。置位时，则其表示屏蔽指令中断时，ARM 内核处于中断屏蔽的边缘。因此，当 SPSR 恢复时，屏蔽指令完成 (中断被屏蔽)。

快速中断

快速中断源

除使用快速强制特性，中断源 0 是唯一能使处理器发出一个快速中断请求的源。中断源 0 可直接或通过 PIO 控制器与产品的 FIQ 引脚连接。

快速中断控制

AIC 的快速中断逻辑没有优先级控制器。中断源 0 模式在 AIC_SMR0 中编程设置，该寄存器的 PRIOR 域即使读取了要写入的值也不会使用。AIC_SMR0 的 SRCTYPE 域使能编程快速中断源为正边沿或负边沿触发；或高电平或低电平敏感。

对 AIC_IECR (中断使能命令寄存器) 与 AIC_IDCR (中断禁用命令寄存器) 写入 0x1，将分别使能和禁用快速中断。 AIC_IMR (中断屏蔽寄存器) 位 0 指示快速中断是否使能。

快速中断向量

快速中断处理程序地址存于 AIC_SVR0 (源向量寄存器 0)。当处理器读 AIC_FVR (快速向量寄存器) 时将返回写入该寄存器中的值。这提供了一个使用单跳转指令到中断处理的方法，由于 AIC_FVR映射在绝对地址0xFFFF F104，通过下面的指令可由地址为0x0000 001C的ARM快速中断向量来访问：

```
LDR   PC,[PC,# -&F20]
```

当处理器执行该指令时，它将由 AIC_FVR 读出的值载入程序计数器，然后跳转到快速中断处理程序。若编程为边沿触发模式，它还自动执行对快速中断源的清除。

快速中断处理程序

该节给出使用 AIC 时快速中断处理序列。假设程序员已了解 ARM 处理器架构，特别是处理器中断模式及相关状态位。

假设如下：

1. 高级中断控制器已编程， AIC_SVR0 寄存器载入快速中断服务程序地址且中断源 0 使能。
2. 位于地址 0x1C (FIQ 异常向量地址) 的指令指向快速中断：
```
LDR PC, [PC, # -&F20]
```
3. 用户不需要嵌套快速中断。

若 CPSR 的位 "F" 为 0， nFIQ 出现，序列为：

1. CPSR 存于 SPSR_fiq，程序计数器当前值载入 FIQ 链接寄存器 (R14_FIQ) 且程序计数器 (R15) 中载入 0x1C。在由地址 0x20 取指周期中， ARM 内核调整 R14_fiq，以 4 递减。
2. ARM 内核进入 FIQ 模式。
3. 当加载地址为 0x1C 的指令执行时，程序计数器载入由 AIC_IVR 读出的值。若编程为边沿触发模式，读 AIC_IVR 将自动清除快速中断。仅在此时，处理器禁用 nFIQ 线。
4. 先前的步骤已使跳转到相应的中断服务程序。若不需要嵌套快速中断，则不用保存链接寄存器 (R14_fiq) 及 SPSR_fiq。
5. 中断处理程序按要求执行。由于 FIQ 自身有专用寄存器且用户 R8 到 R13 为组，因此不需要保存寄存器 R8 到 R13。其它寄存器， R0 到 R7，使用前必须保存并在结束后恢复 (在下一步前)。注意，若快速中断编程为电平敏感，此时必须清除中断源以便释放中断源 0。





6. 最后，链接寄存器 R14_fiq 在减 4 后恢复到 PC 中 (例如，使用指令 `SUB PC, LR, #4`)。这样将返回中断执行前的状态，将 SPSR 中值载入 CPSR 中，是否屏蔽快速中断由存于 SPSR 中的状态决定。

Note: SPSR 中的 "F" 位非常重要。如对其置位，当屏蔽指令中断时表示 ARM 内核屏蔽 FIQ 中断。因此当 SPSR 恢复时，中断指令结束 (FIQ 被屏蔽)。

另一种处理快速中断的方法是将中断服务程序映射到 ARM 向量 0x1C 处。该方法未使用引导向量，因此必须在处理程序开始前读 AIC_FVR。但是，该方法保存跳转指令的执行。

快速强制

高级中断控制器提供的快速强制提供了任意普通中断源在快速中断控制器改向特性。

快速强制的使能与禁用是通过对快速强制使能寄存器 (AIC_FFER) 及快速强制禁用寄存器 (AIC_FFDR) 的写入来实现。对这些寄存器写操作将更新控制每个内部或外部中断源特性的快速强制状态寄存器 (AIC_FFSR)。

当快速强制禁用后中断源处理如前页所述。

当快速强制使能，编程为电平敏感，此时中断源的边沿检测仍然有效，但源无法触发一个普通中断且不能被优先级处理程序处理。

若中断源编程为电平敏感模式且采得有效电平，快速强制结果由 nFIQ 线到内核。

若中断源编程为边沿触发模式且检测到有效边沿，快速强制结果由 nFIQ 线到内核。

快速强制特性不影响中断挂起寄存器 (AIC_IPR) 源 0 挂起位。

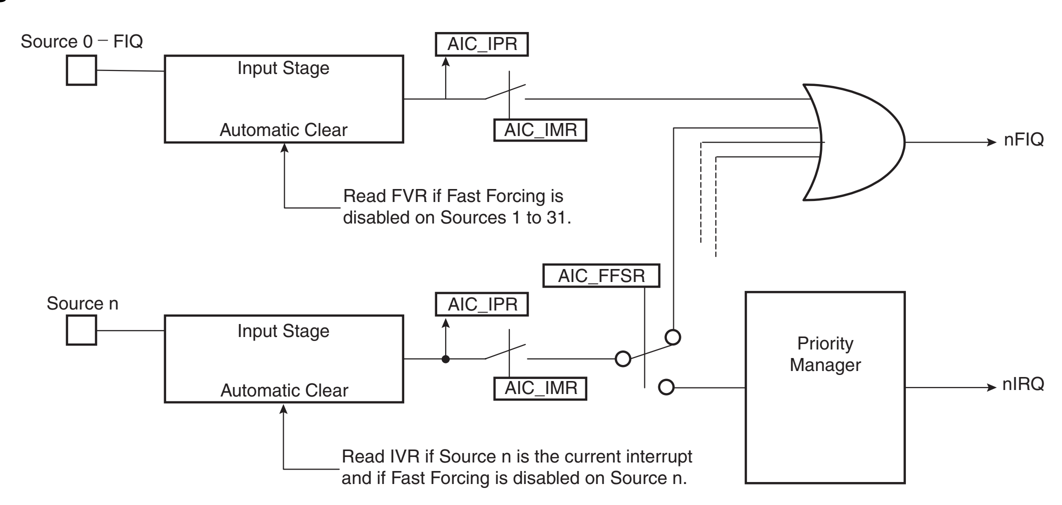
不管是否快速中断源，快速中断向量寄存器 (AIC_FVR) 读源向量寄存器 0 (AIC_SVR0) 中的内容。当使用快速强制特性时，对 FVR 的读取不会将源 0 清除，中断源应通过写中断清除命令寄存器 (AIC_ICCR) 来清除。

所有有快速强制特性使能且编程为边沿触发模式的使能或挂起的中断源必须通过写中断清除命令寄存器来清除。这样，它们将独立清除且防止丢失中断。

读 AIC_IVR 不会清除已使能了快速强制特性的中断源。

保留给快速中断的源 0 维持正常操作，成为一个快速中断源。

Figure 116. 快速强制



保护模式

保护模式允许在不执行相关自动操作时读中断向量寄存器。这对工作在调试系统下的工作来说是必要的。当调试器与调试监控器或 ARM 处理器的 ICE 一起工作时，停止应用程序并更新打开的窗口，可能会读 AIC 用户接口及 IVR。下面是一些不希望见到的结果：

- 若有一个高于当前中断优先级的中断挂起，将其推入堆栈。

- 若没有使能挂起中断，返回先前向量。

上述两种情况下中断结束命令需要得到应答且恢复 AIC 的前后关系。通常该操作不是由调试系统执行，因为调试系统将成为强烈的干扰并引起应用程序进入非理想状态。

这可通过使用保护模式来避免。在 AIC_DCR (调试控制寄存器) 的 DBGM 写 0x1 使能保护模式。

保护模式下，只有当对 AIC_IVR 执行写访问时， AIC 执行中断入栈。因此，中断服务程序在读 AIC_IVR 后必须写入判决数据。AIC 新的前后关系 , 包括中断状态寄存器 (AIC_ISR) 值 , 只有在 AIC_IVR 写入后方能更新当前中断。

AIC_IVR读本身的值(如 ,通过调试器) , 不会改变AIC前后关系及AIC_ISR内容。额外的AIC_IVR 读执行相同的操作。但是，建议在中断服务程序 AIC_IVR 读写间不要停止处理器运行，以保证调试器不修改 AIC 前后关系。

正常操作模式下，在 AIC 中对 AIC_IVR 的读取执行了下列操作：

1. 计算有效中断 (是否高于当前优先级或是假的)。

2. 确定并返回有效中断向量。

3. 存储中断。

4. 将当前优先级推入内部堆栈。

5. 确认中断。

但在保护模式中，读 AIC_IVR 时只需执行 1 到 3 步 ；当写 AIC_IVR 时执行 4 与 5 步。

保护模式下写与调试程序可在正常模式下正确运行而不用加以修改。但是，普通模式下，AIC_IVR 写无影响，因此可将其删除以优化代码。

伪中断

高级中断控制器特性防止伪中断。伪中断的定义是中断源长足以在 AIC 上插入 nIRQ，但当 AIC_IVR 读时已消失。最有可能出现在：

- 外部中断源编程在电平敏感模式下且有效电平仅出现极短的时间。
- 外部中断源编程在电平敏感模式下且内置外设相应的输出信号仅激活极短的时间。
- 软件屏蔽前中断只出现几个周期，因此导致中断源出现脉冲。

AIC 在 AIC_IVR 读且中断源挂起未使能时检测到伪中断 。此时 , AIC 返回值存于 AIC_SPU (伪向量寄存器) 中。程序员在 AIC_SPU 中保存伪中断处理程序地址作为应用程序的一部分，以尽快返回正常执行流程。写入 AIC_EOICR 的处理程序执行中断返回。

通用中断屏蔽

AIC 的通用中断屏蔽位防止中断传递到处理器。若 AIC_DCR (调试控制寄存器) 中的 GMSK 位置位， nIRQ 与 nFIQ 线将驱动到它们的停止状态。但是该屏蔽不妨碍处理器由空闲模式下的唤醒。该功能方便处理器在下一事件上同步 , 当事件出现后不用处理中断 , 立即执行并发操作。建议小心使用该屏蔽。



高级中断控制器 (AIC) 用户接口

基地址 AIC映射到地址**0xFFFF F000**上。共有4-K字节的地址空间。因ARM处理器与PC相关的下载/保存指令仅支持 ± 4-K 字节的偏移，因此允许向量引导特性。

Table 59. 寄存器映射

偏移	寄存器	名称	访问	复位值
0000	源模式寄存器 0	AIC_SMR0	读 / 写	0x0
0x04	源模式寄存器 1	AIC_SMR1	读 / 写	0x0
–	–	–	–	–
0x7C	源模式寄存器 31	AIC_SMR31	读 / 写	0x0
0x80	源向量寄存器 0	AIC_SVR0	读 / 写	0x0
0x84	源向量寄存器 1	AIC_SVR1	读 / 写	0x0
–	–	–	–	–
0xFC	源向量寄存器 31	AIC_SVR31	读 / 写	0x0
0x100	中断向量寄存器	AIC_IVR	只读	0x0
0x104	快速中断向量寄存器	AIC_FVR	只读	0x0
0x108	中断状态寄存器	AIC_ISR	只读	0x0
0x10C	中断挂起寄存器	AIC_IPR	只读	0x0(1)
0x110	中断屏蔽寄存器	AIC_IMR	只读	0x0
0x114	内核中断状态寄存器	AIC_CISR	只读	0x0
0x118	保留	–	–	–
0x11C	保留	–	–	–
0x120	中断使能命令寄存器	AIC_IECR	只写	–
0x124	中断禁用命令寄存器	AIC_IDCR	只写	–
0x128	中断清除命令寄存器	AIC_ICCR	只写	–
0x12C	中断置位命令寄存器	AIC_ISCR	只写	–
0x130	中断结束命令寄存器	AIC_EOICR	只写	–
0x134	伪中断向量寄存器	AIC_SPU	读 / 写	0x0
0x138	调试控制寄存器	AIC_DCR	读 / 写	0x0
0x13C	保留	–	–	–
0x140	快速强制使能寄存器	AIC_FFER	只写	–
0x144	快速强制禁用寄存器	AIC_FFDR	只写	–
0x148	快速强制状态寄存器	AIC_FFSR	只读	0x0

Note: 1. 中断挂起寄存器复位值由外部中断源电平决定。其它所有中断源在复位时清零而不挂起。

AIC 源模式寄存器

寄存器名称 : AIC_SMR0..AIC_SMR31

访问类型 : 读 / 写

复位值 : 0x0

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	SRCTYPE		–	–	PRIOR		

- **PRIOR: 优先级**

对除 FIQ 源 (源 0) 外的中断源优先级编程。

优先级在 0 (最低) 到 7 (最高) 之间。

优先级没有在与 SMR 寄存器 AIC_SMRx 相关的 FIQ 中使用。

- **SRCTYPE: 中断源类型**

内部中断源有效电平或边沿不可编程。

SRCTYPE		内部中断源
0	0	电平敏感
0	1	边沿触发
1	0	电平敏感
1	1	边沿触发

AIC 源向量寄存器

寄存器名称 : AIC_SVR0..AIC_SVR31

访问类型 : 读 / 写

复位值 : 0x0

31	30	29	28	27	26	25	24
VECTOR							

23	22	21	20	19	18	17	16
VECTOR							

15	14	13	12	11	10	9	8
VECTOR							

7	6	5	4	3	2	1	0
VECTOR							

- **VECTOR: 源向量**

用户可在这些寄存器中存储各个中断源相关处理程序地址。

AIC 中断向量寄存器

寄存器名称 : AIC_IVR

访问类型： 读 / 写

复位值： 0

31	30	29	28	27	26	25	24
IRQV							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
IRQV							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
IRQV							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
IRQV							

- **IRQV: 中断向量寄存器**

中断向量寄存器包含了用户编程的当前中断相关的源向量寄存器。

读中断向量寄存器时，源向量寄存器使用中断号码作为索引。

当没有当前中断，中断向量寄存器读取存储于 AIC_SPU 中的值。

AIC FIQ 向量寄存器

寄存器名称： AIC_FVR

访问类型： 只读

复位值： 0

31	30	29	28	27	26	25	24
FIQV							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
FIQV							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
FIQV							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FIQV							

- **FIQV: FIQ 向量寄存器**

FIQ向量寄存器包含用户在源向量寄存器0中编程的向量值。当无快速中断时，快速中断向量寄存器读存于AIC_SPU中的值。

AIC 中断状态寄存器

寄存器名称： AIC_ISR

访问类型： 只读

复位值： 0

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	–	IRQID				

- **IRQID: 当前中断标识**

中断状态寄存器返回当前中断源序号。

AIC 中断挂起寄存器

寄存器名称： AIC_IPR

访问类型： 只读

复位值： 0

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PID7	PID6	PID5	PID4	PID3	PID2	SYS	FIQ

- **FIQ, SYS, PID2-PID31: 中断挂起**

0 = 相关中断未挂起。

1 = 相关中断挂起。



AIC 中断屏蔽寄存器

寄存器名称： AIC_IMR

访问类型： 只读

复位值： 0

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PID7	PID6	PID5	PID4	PID3	PID2	SYS	FIQ

- **FIQ, SYS, PID2-PID31: 中断屏蔽**

0 = 相应中断禁用。

1 = 相应中断使能。

AIC 内核中断状态寄存器

寄存器名称： AIC_CISR

访问类型： 只读

复位值： 0

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	–	–	–	–	NIRQ	NIFQ

- **NFIQ: NFIQ 状态**

0 = nFIQ 线禁用。

1 = nFIQ 线激活。

- **NIRQ: NIRQ 状态**

0 = nIRQ 线禁用。

1 = nIRQ 线激活。

AIC 中断使能命令寄存器

寄存器名称： AIC_IECR

访问类型： 只写

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	FIQ

- **FIQ, SYS, PID2-PID3: 中断使能**

0 = 无效。

1 = 使能相应中断。

AIC 中断禁用命令寄存器

寄存器名称： AIC_IDCR

访问类型： 只写

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	FIQ

- **FIQ, SYS, PID2-PID31: 中断禁用**

0 = 无效。

1 = 禁用相应中断。

AIC 中断清除命令寄存器

寄存器名称： AIC_ICCR

访问类型： 只写

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24

23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16

15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8

7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	FIQ

- **FIQ, SYS, PID2-PID31: 中断清除**

0 = 无效。

1 = 清除相应中断。

AIC 中断置位命令寄存器

寄存器名称：AIC_ISCR

访问类型： 只写

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24

23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16

15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8

7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	FIQ

- **FIQ, SYS, PID2-PID31: 中断置位**

0 = 无效。

1 = 置位相应中断。

AIC 中断结束命令寄存器

寄存器名称： AIC_EOICR

访问类型： 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	–	–	–	–	–	–

中断处理程序用中断结束命令寄存器指示中断处理结束。因为只需要向该寄存器地址写值以标识中断处理结束，所以可向其写入任意值。

AIC 伪中断向量寄存器

寄存器名称： AIC_SPU

访问类型：读 / 写

复位值： 0

31	30	29	28	27	26	25	24
SIQV							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
SIQV							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
SIQV							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SIQV							

- **SIQV: 伪中断向量寄存器**

用户可在该寄存器中存储伪中断处理程序的地址。当出现未中断时，写入该寄存器的值为 AIC_IVR 的返回值，类似的发生伪快速中断时，写入该寄存器的值为 AIC_IVR 的返回值。



AIC 调试控制寄存器

寄存器名称 : AIC_DEBUG

访问类型 : 读 / 写

复位值 : 0

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	GMSK	PROT

- **PROT: 保护模式**

0 = 保护模式禁用。

1 = 保护模式使能。

- **GMSK: 产生屏蔽**

0 = nIRQ 与 nFIQ 线由 AIC 控制。

1 = nIRQ 与 nFIQ 线置于无效状态。

AIC 快速强制使能寄存器

寄存器名称 : AIC_FFER

访问类型 : 只写

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	–

- **SYS, PID2-PID31: 快速强制使能**

0 = 无效。

1 = 使能相应中断快速强制特性。

AIC 快速强制禁用寄存器

寄存器名称 : AIC_FFDR

访问类型 : 只写

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	–

- **SYS, PID2-PID31: 快速强制禁用**

0 = 无效。

1 = 禁用相应中断快速强制特性。



AIC 快速强制状态寄存器

寄存器名称：AIC_FFSR

访问类型：只读

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24

23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16

15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8

7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	–

- **SYS, PID2 - PID31: 快速强制状态**

0 = 禁用相应中断的快速强制特性。

1 = 使能相应中断的快速强制特性。

电源控制器 (PMC)

概述

电源控制器 (PMC) 集成了了两个振荡器和两个 PLL，产生系统所有的时钟。

PMC 向内置处理器提供时钟，停止处理器时钟进入空闲模式直到下一个中断到来。

PMC独立提供和控制多达30个外设时钟及4 个作为输出接到引脚以支持外部设备的可编程时钟。集成的 PLL 以总线速度要求供给 USB 器件和主机端口 48 MHz 的时钟，及供给其它部分系统另外的频率。因此，全功能电源管理控制器优化整个系统功耗，并支持普通、空闲、慢时钟及 Standby 工作模式。

PMC 主要特性如下：

- 优化整个系统功耗
- 内置与控制：
  - 一个主振荡器及一个慢时钟振荡器 (32.768 kHz)
  - 两个锁相环 (PLLs) 及分频器
  - 时钟预定标器
- 提供：
  - 处理器时钟 PCK
  - 主机时钟 MCK
  - USB 时钟， UHPCK 与 UDPCK，分别供给 USB 主机端口及 USB 器件端口
  - 在 USB 挂起状态下的可编程自动 PLL 断开
  - 多达 30 个的外设时钟
  - 多达 4 个的可编程时钟输出
- 4 种工作模式：
  - 普通模式、空闲模式、慢时钟模式、 Standby 模式

附属产品

I/O 线

电源管理控制器可处理高达 4 个可编程时钟， PCK0 到 PCK3。

可编程时钟通常与 PIO 控制器复用。必须先对 PIO 控制器编程，将可编程时钟引脚分配到其外设功能上。

中断

电源管理控制器有一条与高级中断控制器 (AIC) 连接的中断线。处理 PMC 中断请求对 AIC 编程前要先配置 PMC。

振荡器与 PLL 特性

内置振荡器与 PLL 的电气特性是与产品相关的，即使控制它们的方式类似。

产品手册的 DC 特性部分给出振荡器与 PLL 的所有特性。这些图表不仅在硬件使用中使用，因为它们影响到连接在引脚的外部组件，而且在软件配置中也需要用到它们，因为它们决定了启动与锁定时间编程的等待时间。

外设时钟

电源管理控制器提供与控制高达 30 个外设时钟。外设时钟允许控制的位序号为内置外设的外设 ID 编号。

当外设 ID 没有对应外设，不管是因为有一个外设中断还是 因为系统少于 30 个外设，对该外设 ID 的控制位不会在 PMC 中执行，对它们的编程也不影响 PMC 的工作。

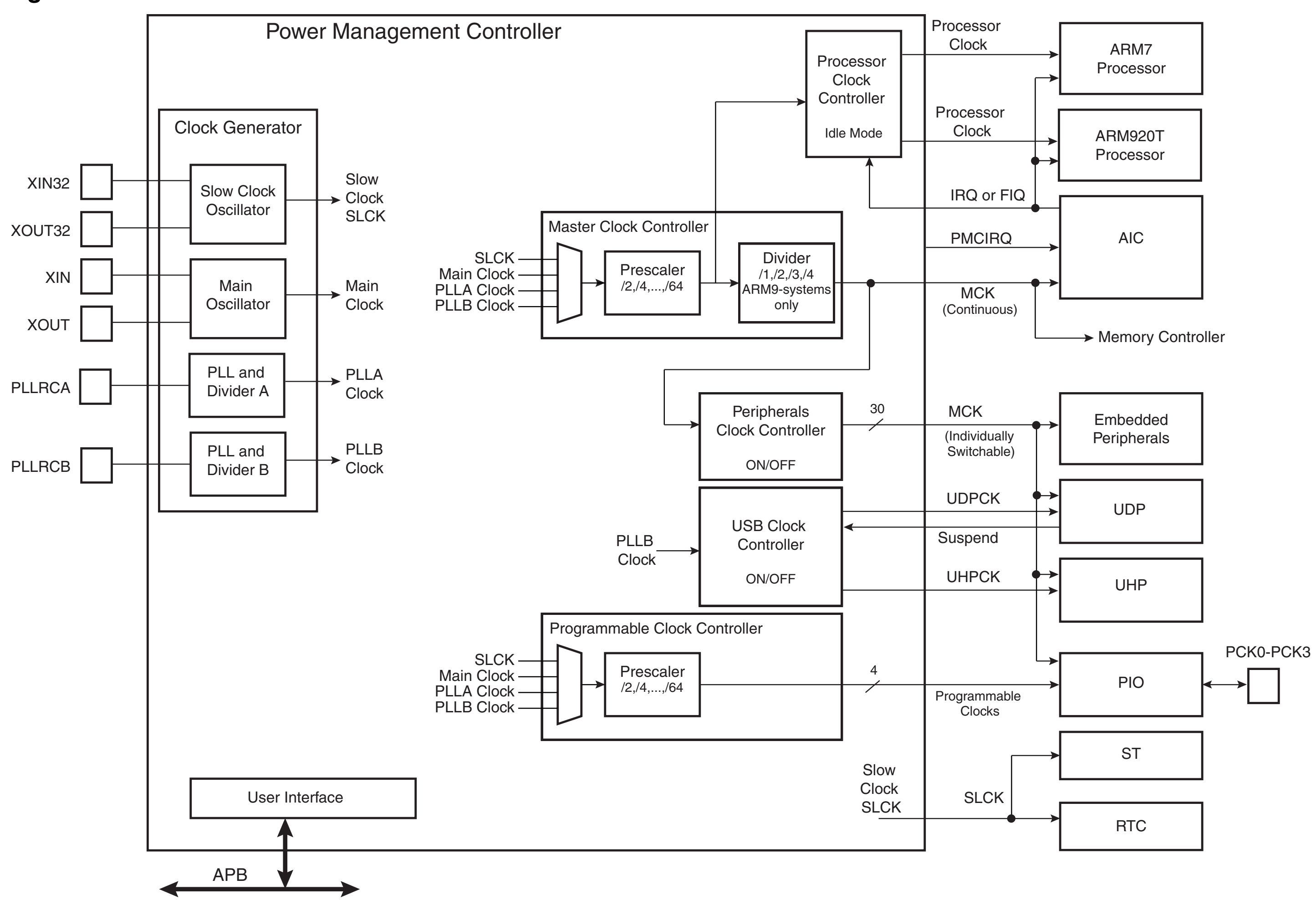
USB 时钟

电源管理控制器提供并控制两个 USB 时钟，对于 USB 主机端口的 UHPCK 及对于 USB 器件的 UDPCK 时钟。

若产品既没有内置 USB 主机端口也没有内置 USB 器件端口 ,则相关的控制位与寄存器不会在 PMC 中执行，对它们的编程也不会影响 PMC 工作。

方框图

Figure 117. 电源管理控制器框图



功能说明

工作模式定义

PMC 支持下列工作模式，并提供不同的功耗等级及事件响应延迟时间：

- 普通模式：ARM 处理器时钟时钟使能，外设时钟是否使能由应用需求决定。
- 空闲模式:ARM 处理器时钟禁用并等待下一次中断 (或主复位)。外设时钟是否使能由应用需求决定。可能需要 PDC 传输。
- 慢时钟模式 : 慢时钟模式与普通模式类似 , 但主振荡器及 PLL 关闭以节约功耗 , 处理器与外设运行在慢时钟模式下。注意，慢时钟模式是复位后选择的模式。
- Standby 模式：Standby 模式是慢时钟模式与空闲模式下结合。它使能处理器以快速响应唤醒事件，并能保持较低的功耗。

时钟定义

电源管理控制器提供下列时钟：

- 慢时钟 (SLCK)，系统内唯一不变的时钟，频率为 32.768 kHz。
- 主机时钟 (MCK)，编程范围从几百 Hz 到器件最大工作频率。它有利于模块的长期运行，如 AIC 及存储控制器。
- 处理器时钟 , 基于 ARM7 系统的主机时钟 , 基于 ARM9 系统的快速时钟 , 当进入空闲模式时关闭。
- 外设时钟，典型频率为 MCK 频率，提供给内置外设 (USART、 SSC、 SPI、 TWI、TC、 MCI 等) 并独立可控。为减少产品时钟名称数目，在产品手册中外设时钟称为 MCK。
- UDP 时钟 (UDPCK)，典型频率为 48 MHz，用于 USB 器件端口工作。
- UHP 时钟 (UHPCK)，典型频率为 48 MHz，用于 USB 主机端口工作。
- 可编程时钟输出 (PCK0 到 PCK3) 可从时钟发生器提供的时钟中选择 , 并在 PCK0 到 PCK3 引脚上驱动。

时钟发生器

时钟发生器内置：

- 慢时钟振荡器
- 主振荡器
- 两个 PLL 及分频器模块， A 与 B

时钟发生器可选集成一个 2 倍分频器。基于 ARM7 的系统通常内置的 PLL 可输出频率范围为 20 MHz 到 100 MHz , 且没有内置的 2 倍分频器。基于 ARM9 的系统通常内置的 PLL 可输出频率范围为 80 MHz 到 240 MHz。由于这是 PLL 不能达到 USB 所需的 48 MHz，则用到 2 倍分频器。

时钟发生器框图见 Figure 118。

Figure 118. 时钟发生器框图

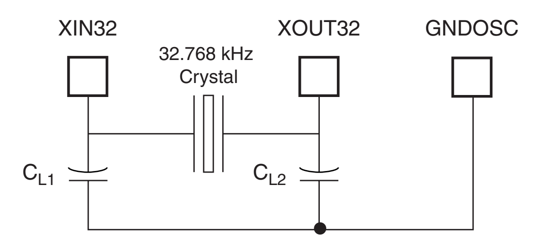
Clock Generator
XIN32
XOUT32
XIN
XOUT
PLLRCA
PLLRCB
Slow Clock Oscillator
Main Oscillator
Main Clock Frequency Counter
Divider A
PLL A
Divider B
PLL B
/2 (optional)
Slow Clock SLCK
Main Clock
PLLA Clock
PLLB Clock

慢时钟振荡器

慢时钟振荡器连接

时钟振荡器连接一个低功耗的 32.768 kHz 振荡器。XIN32 与 XOUT32 引脚必须与 32.768 kHz 晶振连接。两个电容的连接如 Figure 119 所示。

Figure 119. 典型的慢时钟振荡器连接



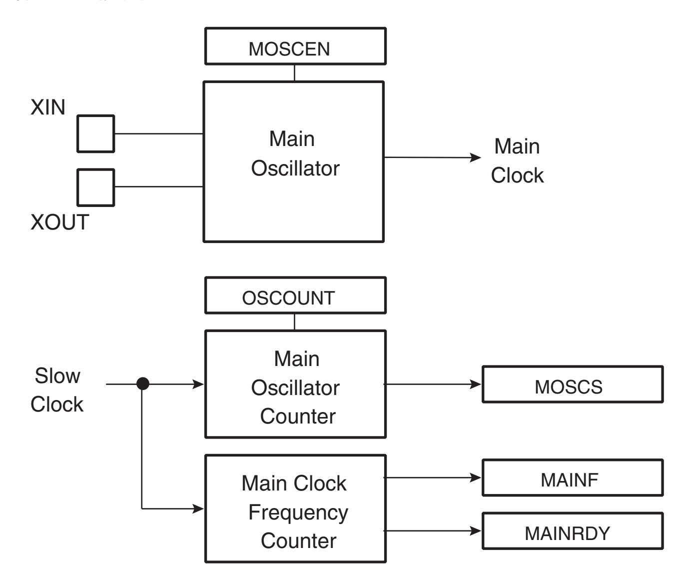
慢时钟振荡器启动时间

产品手册的 DC 特性部分给出了慢时钟振荡器的启动时间。由于该启动时间高于 500 ms，且处理器需要在其稳定后出现复位，因此用户必须执行一个外部复位监管来覆盖该启动时间。但，该启动时间仅出现在冷复位中，即系统上电时。当热复位出现时，复位脉冲长度可能很短，详见 “AT91RM9200 Reset Controller” on page 119。

主振荡器

Figure 120 给出主振荡器框图。

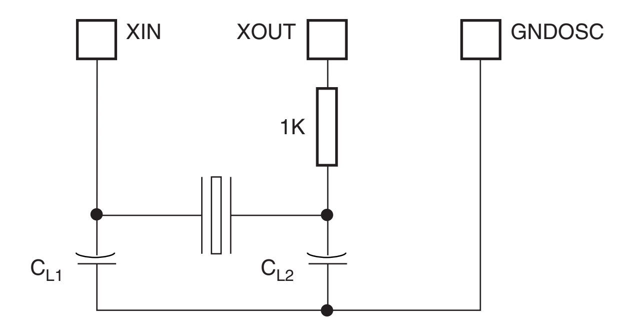
Figure 120. 主振荡器框图



主振荡器连接

时钟发生器集成了一个基本频率为3到20 MHz的主机振荡器晶体。典型晶体连接见Figure 121。只有当频率低于 8 MHz 时需要 1 kΩ 电阻。振荡器在每个 XIN 与 XOUT 引脚上连接 25pF 的电容。因此，当晶体负载电容为 12.5 pF 时，可将 CL1 与 CL2 移走。关于主振荡器更多的内容见产品手册的 DC 特性部分。

Figure 121. 典型的晶体连接



主振荡器启动时间

产品手册 DC 特性部分给出主振荡器启动时间。启动时间取决于晶体频率及频率升高幅度。

主振荡器控制

为最小化系统启动功耗，复位后主振荡器禁用并选择慢时钟模式。

软件通过清除主振荡器寄存器 (CKGR_MOR) 的 MOSCEN 位来禁用主振荡器，以降低功耗。当通过清除 CKGR_MOR 的 MOSCEN 位将主振荡器禁用后，PMC_SR 中的 MOSCS 位自动清零，表示主时钟关闭。

使能主振荡器时，用户必须用振荡器相应的启动时间值初始化主振荡器计数器。该启动时间由与主振荡器连接的晶体频率决定。当 CKGR_MOR 寄存器中 MOSCEN 位与 OSCOUNT 位写入时使能主振荡器，MOSCS 位清零，计数器开始以慢时钟 8 分频的速率从 OSCOUNT 值开始向下计数。由于 OSCOUNT 值为 8 位，所以最大的启动时间约为 62 ms。

当计数器达到 0， MOSCS 位置位，表示主时钟有效。设置 PMC_IMR 中的 MOSCS 位可触发该中断事件。

主时钟频率计数器

主振荡器的主时钟频率计数器提供与主振荡器连接的石英频率。通常，仅系统设计者知道该值，但它对引导程序以正确的时钟速率配置器件非常有用。

主时钟频率计数器在主振荡器稳定后，即 MOSCS 置位后的下一个慢时钟上升沿处以主时钟速度开始递增。然后在慢时钟第 16 个下降沿处， CKGR_MCFR（主时钟频率寄存器）的 MAINRDY 位置位，计数器停止计数。其值可在 CKGR_MCFR 的 MAINF 域读出，并给出 16 个慢时钟周期中的主时钟周期数，这样可得到与主振荡器连接的晶体频率。

主振荡器旁路

用户向器件提供时钟可不通过连接晶体。此时，用户必须在 XIN 引脚提供外部时钟信号。此时 XIN 引脚输入的符号见产品电气特性部分。程序员必须保证不修改 CKGR_MOR 寄存器中的 MOSCEN 位。该位必须保持为 0，以保证外部时钟正常工作。当该位为 0，XIN 引脚拉低以防止内部振荡。

外部时钟信号必须满足相关电源供给 VDDPLL（即在 1.65V 到 1.95V 间），不能超过 50 MHz。

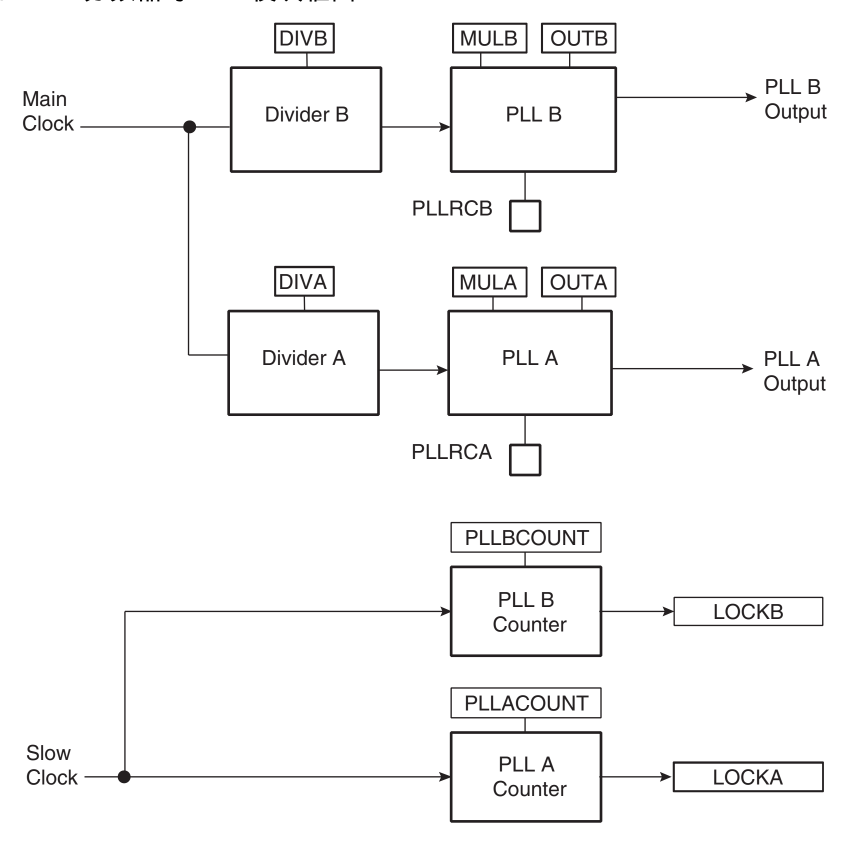


分频器与 PLL 模块

时钟发生器的两个分频器 /PLL 块产生宽频率。此外，不管主时钟，向内置的 USB 器件与 / 或主机端口提供 48 MHz 信号。

Figure 122 给出分频器与 PLL 模块方框图。

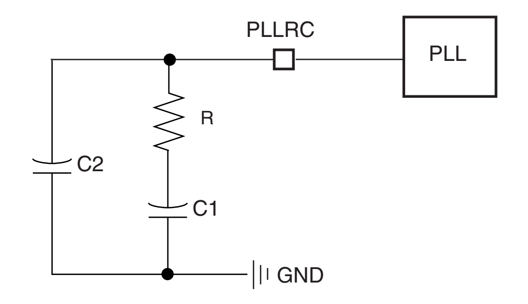
Figure 122. 分频器与 PLL 模块框图



PLL 滤波器

PLL 需要通过 PLLRC 引脚与一个外部二阶滤波器连接。 Figure 123 给出滤波器示意图。

Figure 123. PLL 电容与电阻



与 PLLRC 连接的 R、 C1 与 C2 值由 PLL 输入频率、 PLL 输出频率及相位容限确定。必须在输出信号过量与启动时间间找到平衡。

PLL 源时钟

PLL 源 A 与 B 分别为分频器 A 输出，即主时钟由 DIVA 分频；为分频器 B 输出，即主时钟由 DIVB 分频。

由于 PLL 输入频率受限，用户必须保证对 DIVA 与 DIVB 编程适合产品手册 DC 特性中给定的 PLL 的输入范围。

分频器与锁相环编程

两个分频器增加独立于输入频率的 PLLA 与 PLLB 时钟精度。

可通过对 CKGR_PLLBR 的 DIVB 域及 CKGR_PLLAR 的 DIVA 域编程对主时钟分频。分频器可在 1 到 255 间置位，步长为 1。当 DIVA 与 DIVB 域设置为 0，分频器及 PLL 输出为连续的 0 信号。复位时，每个 DIV 域置为 0，因此相应的 PLL 输入时钟置为 0。

PLL 允许分频器输出相乘。PLL 时钟信号频率由各自源信号频率及参数 DIV (DIVA、DIVB) 与 MUL (MULA、 MULB) 确定。源信号频率系数为 (MUL + 1)/DIV。当 MUL 写入 0，相应的 PLL 禁用并节省其功耗。在 MUL 域写入大于 0 的值将重新使能 PLL。

当 PLL 重新使能或它的某个参数改变，PMC_SR 中的 LOCK 位自动清零。CKGR_PLLR 中 PLLCOUNT 域值载入 PLL 计数器。 PLL 计数器开始以慢时钟速率开始递减直到其值为 0。此时，LOCK 位置位并能触发处理器中断。用户须在 PLLCOUNT 域载入所需慢时钟周期数来覆盖 PLL 过渡时间。过渡时间由 PLL 滤波器确定。PLL 初始状态及其目标频率可使用 Atmel 提供的专用工具进行计算。

PLLB 2 分频

基于 ARM9 内核系统， PLLB 时钟可被 2 分频。该分频器可通过置位 CKGR_PLLBR 中的 USB_96M 位使能。该情况下，2 倍分频器使能，且 PLLB 必须编程输出 96 MHz 而非 48 MHz，以确保 USB 总线正确工作。

时钟控制器

电源管理控制器给系统的不同外设提供时钟。它内置下列元素：

- 主机时钟控制器，用来选择主机时钟。
- 处理器时钟控制器，用来执行空闲模式。
- 外设时钟控制器，通过控制内置外设时钟来节省功耗。
- USB 时钟控制器，向 USB 控制器分配 48 MHz 时钟。
- 可编程时钟控制器，允许产生多达 4 个外部引脚的可编程时钟信号。

主机时钟控制器

主机时钟控制器提供主机时钟(MCK)的选择与分频。MCK为所有外设及存储控制器时钟。

主机时钟由时钟发生器提供的时钟中选择。选择慢时钟则向整个器件提供一个慢时钟信号。选择主时钟会节省 PLL 功耗，但不允许使用 USB 端口。选择 PLLB 时钟可节省 PLLA 功耗，USB 端口运行处理器及外设在 48 MHz。运行 USB 端口在 48 MHz 时，选择 PLLA 时钟运行处理器及外设在最大速度下。

主机时钟控制器由时钟选择器及预分频器组成，见 Figure 124。集成了 ARM9 处理器的产品中包括一个可选的主机时钟分频器。这使得处理器时钟比主机时钟更快。

通过写 PMC_MCKR (主机时钟寄存器) 的 CSS 域 (时钟源选择) 对主机时钟进行选择。预分频器分频因子为在 1 到 64 间 2 的幂次。PMC_MCKR 中的 PRES 域对预分频器编程。

执行主机时钟分频时，可通过编程 PMC_MCKR 中的 MDIV 域设置分频值在 1 到 4 之间。

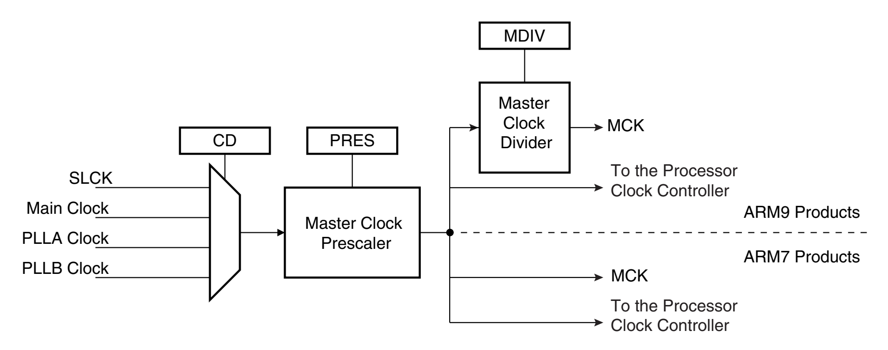
每次 PMC_MCKR 写入定义一个新的主机时钟， PMC_SR 中的 MCKRDY 位清除。在主机时钟建立前它的值为 0。然后， MCKRDY 位被置位并能触发处理器中断。该特性在当由高速时钟向低速时钟切换，用来通知何时改变。

Note: 写入 PMC_MCKR 的新值不许与 PMC_MCKR 中的当前值相同。





Figure 124. 主机时钟控制器



处理器时钟控制器

PMC 是使处理器时钟控制器 (PCK) 执行处理器空闲模式。通过写系统时钟使能寄存器 (PMC_SCER) 及系统时钟禁用寄存器 (PMC_SCDR) 可使能或禁用处理器时钟。该时钟状态可在系统时钟状态寄存器 (PMC_SCSR) 中读取。

处理器时钟源

主机时钟控制器决定处理器时钟。在基于 ARM7 内核的系统中，处理器时钟源就是主机时钟。

在基于 ARM9 内核的系统中，处理器时钟可以为 2、3 或 4 倍主机时钟。该比率值通过对主机时钟寄存器 (PMC_MCKR) 的 MDIV 域编程决定。

空闲模式

处理器时钟在复位后使能，在任意使能中断后自动重使能。通过禁用处理器时钟进入空闲模式。该模式下，可通过任意快速或正常中断，或产品复位来自动重使能进入。

当处理器时钟禁用，当前指令在时钟停止前结束，但不会影响其它主机到系统总线的数据传输。

外设时钟控制器

PMC 控制每个内置外设时钟。用户可通过对外设时钟使能 (PMC_PCER) 及外设时钟禁用 (PMC_PCDR) 寄存器来独立使能或禁用外设的主机时钟。在外设时钟状态寄存器 (PMC_PCSR) 中可读取外设时钟状态。

当外设时钟禁用时，该时钟立即停止。当外设时钟重新使能，外设恢复到停止前的状态。外设时钟在复位后自动禁用。

停止外设时，建议在禁用时钟前执行完最后一条指令。以避免数据出错或系统出错。

外设时钟控制寄存器位序号 (PMC_PCER、 PMC_PCDR 及 PMC_PCSR) 为定义在产品级的外设标识符。通常，该序号对应于分配给外设的中断源序号。

USB 时钟控制器

若使用USB端口，用户必须对分频器及PLL B块编程输出精度为± 0.25%的48 MHz信号。

当 USB 时钟稳定， USB 器件与主机时钟 UDPCK 与 UHPCK 可被使能。当 USB 处理结束后将其禁用，这样可节省 48 MHz 信号在这些外设上产生的功耗。

USB 端口同时需要 48 MHz 信号及主机时钟。通过外设时钟控制器来控制主机时钟。

USB 器件时钟控制

通过在PMC_SCER (系统时钟使能寄存器)的UDP位写1使能USB器件端口时钟UDPCK；在 PMC_SCDR (系统时钟禁用寄存器) 的 UDP 位写 1 将禁用 UDPCK。UDPCK 状态见 PMC_SCSR (系统时钟状态寄存器) 的 UDP 位。

USB 器件端口挂起

当 USB 器件端口检测到挂起状态，自动禁用 48 MHz 时钟，即清除 PMC_SCSR 寄存器的 UDP 位。在挂起状态时也可自动禁用供向 USB 器件端口的主机时钟。PMC_SCSR 中的 MCKUDP 位配置该特性，并可在 PMC_SCER 或 PMC_SCDR 的相同 UDP 位写 1 来设置或清除该特性。

USB 主机时钟控制

USB主机端口时钟UHPCK可通过在PMC_SCER (系统时钟使能寄存器)的UHP位写1使能，在PMC_SCDR (系统时钟禁用寄存器)UHP位写1禁用。UDPCK状态见PMC_SCSR (系统时钟状态寄存器) 的 UHP 位。

可编程时钟输出控制器

PMC 控制外部引脚 PCK0 到 PCK3 的 4 个输出信号。每个信号均可通过 PMC_PCK0 到 PMC_PCK3 寄存器独立编程。

PCK0到PCK3可通过在PMC_PCK0到PMC_PCK3的CSS域写入独立选择由时钟发生器提供的 4 个时钟。每个输出信号可由 1 到 64 间 2 的幂次进行分频，具体系数在 MC_PCK0 到 PMC_PCK3 中的 PRES (预分频) 域写入。

通过在 PCKx 中 PMC_SCER 或 PMC_SCDR 位写入 1 来使能或禁用输出信号。工作的可编程时钟状态由 PMC_SCSR(系统时钟状态寄存器) 中的 PCKx 位给出。

此外，与 PCK 类似， PMC_SR 中的状态位表示可编程时钟实际上是对可编程时钟寄存器的什么编程。

由于可编程时钟控制器不会管理当切换时钟时出现的脉冲，强烈建议在配置变化前将可编程时钟禁用并在变化后重新使能。

还要注意需要给 PIO 控制器的可编程时钟操作配置引脚以使能在引脚上的驱动信号。



时钟切换

主机时钟切换时间

Table 60 给出主机时钟切换最差情况下的时间需求。此时预分频器未激活。当预分频器激活，增加新选择时钟的 64 个时钟周期。

Table 60. 时钟切换时间 (最差情况)

从 到	主时钟	SLCK	PLLA 时钟	PLLB 时钟
主时钟	–	4 x SLCK + 2.5 x 主时钟	3 x PLLA 时钟 + 4 x SLCK + 1 x 主时钟	3 x PLLB 时钟 + 4 x SLCK + 1 x 主时钟
SLCK	0.5 x 主时钟 + 4.5 x SLCK	–	3 x PLLA 时钟 + 5 x SLCK	3 x PLLB 时钟 + 5 x SLCK
PLLA 时钟	0.5 x 主时钟 + 4 x SLCK + PLLACOUNT x SLCK + 2.5 x PLLA 时钟	2.5 x PLLA 时钟 + 5 x SLCK + PLLACOUNT x SLCK	2.5 x PLLA 时钟 + 4 x SLCK + PLLB COUNT x SLCK	3 x PLLA 时钟 + 4 x SLCK + 1.5 x PLLA 时钟
PLLB 时钟	0.5 x 主时钟 + 4 x SLCK + PLLBCOUNT x SLCK + 2.5 x PLLB 时钟	2.5 x PLLB 时钟 + 5 x SLCK + PLLBCOUNT x SLCK	3 x PLLB 时钟 + 4 x SLCK + 1.5 x PLLB 时钟	2.5 x PLLB 时钟 + 4 x SLCK + PLLACOUNT x SLCK

时钟切换波形

Figure 125. 主机时钟由慢时钟切换到 PLLA 时钟

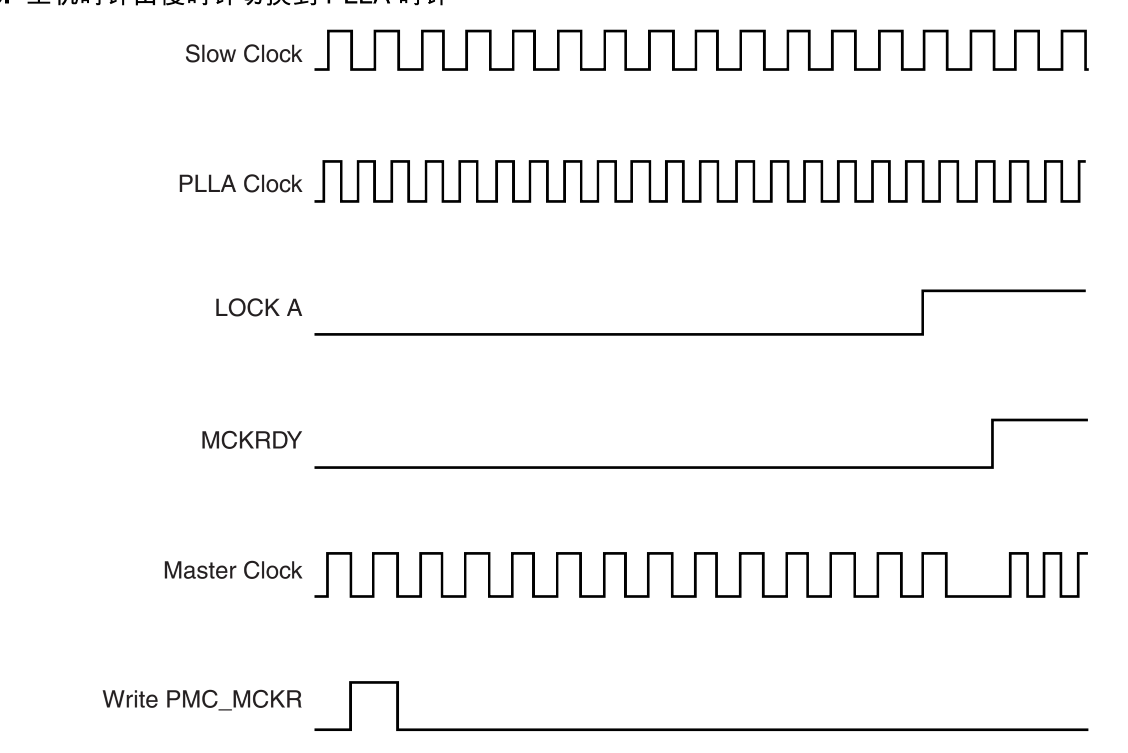


Figure 126. 主机时钟由主时钟切换到慢时钟

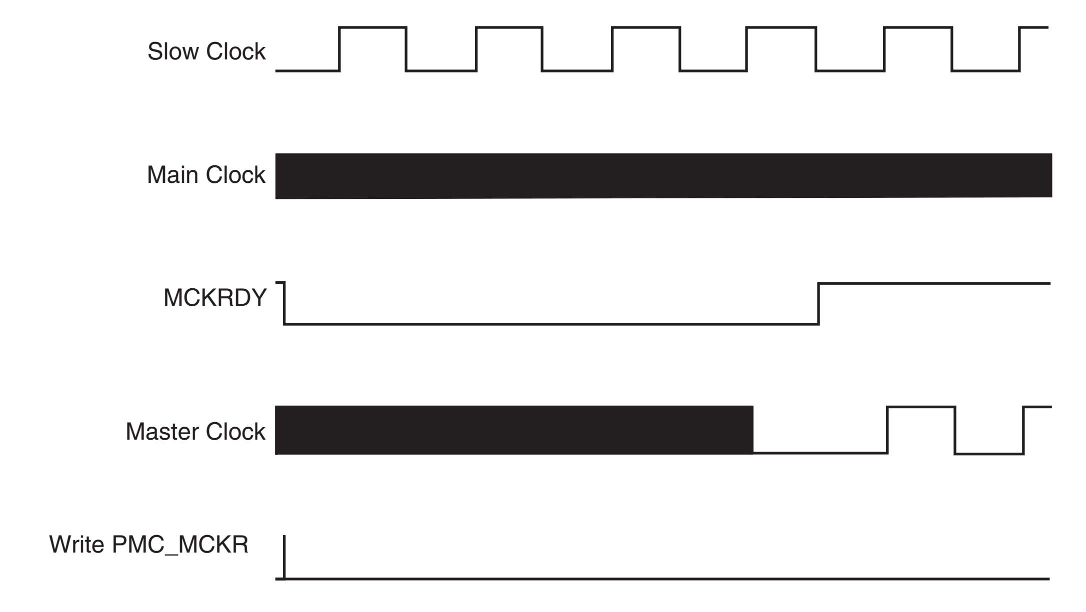


Figure 127. 改变 PLLA 编程

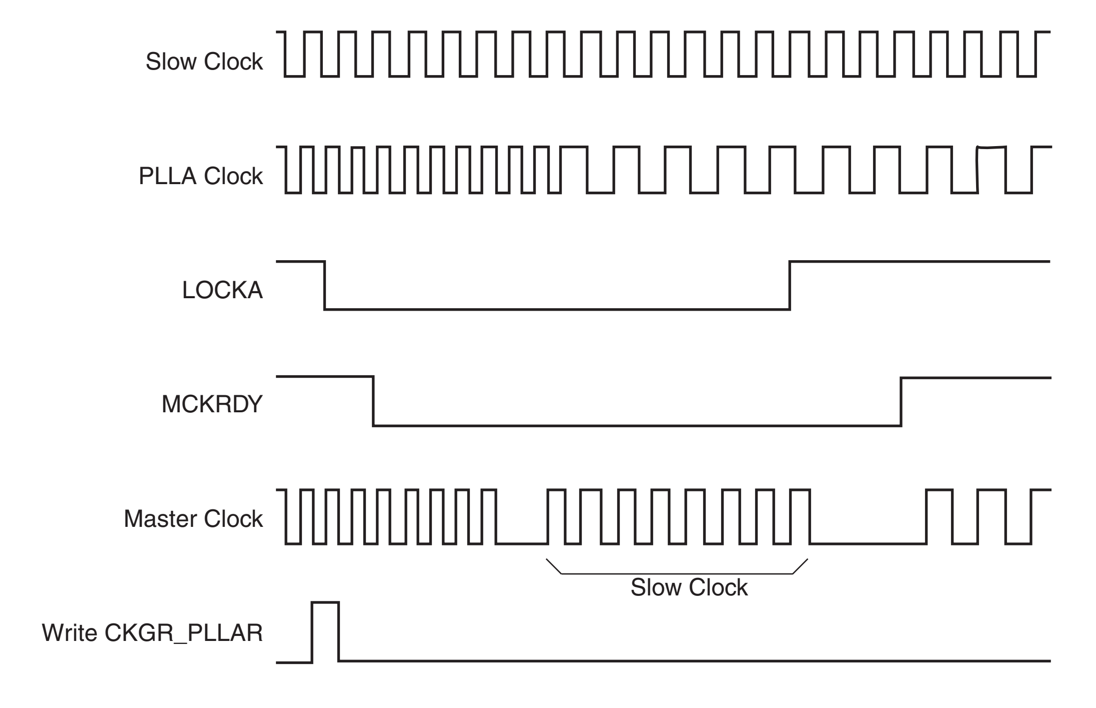
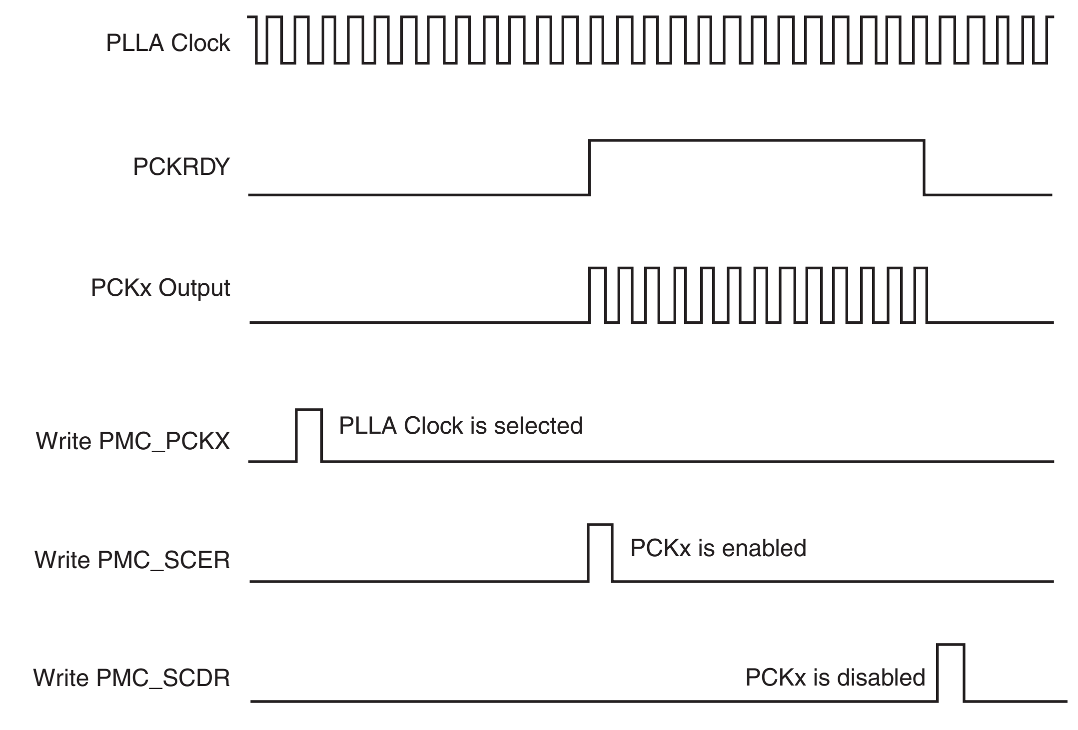


Figure 128. 可编程时钟输出编程



电源管理控制器 (PMC) 用户接口

Table 61. 寄存器映射

偏移	寄存器	名称	访问类型	复位值
0x0000	系统时钟使能寄存器	PMC_SCER	只写	–
0x0004	系统时钟禁用寄存器	PMC_SCDR	只写	–
0x0008	系统时钟状态寄存器	PMC _SCSR	只读	0x01
0x000C	保留	–	–	–
0x0010	外设时钟使能寄存器	PMC _PCER	只写	–
0x0014	外设时钟禁用寄存器	PMC_PCDR	只写	–
0x0018	外设时钟状态寄存器	PMC_PCSR	只读	0x0
0x001C	保留	–	–	–
0x0020	主振荡器寄存器	CKGR_MOR	读 / 写	0x0
0x0024	主时钟频率寄存器	CKGR_MCFR	只读	-
0x0028	PLL A 寄存器	CKGR_PLLAR	读 / 写	0x3F00
0x002C	PLL B 寄存器	CKGR_PLLBR	读 / 写	0x3F00
0x0030	主机时钟寄存器	PMC_MCKR	读 / 写	0x00
0x0034	保留	–	–	–
0x0038	保留	–	–	–
0x003C	保留	–	–	–
0x0040	可编程时钟 0 寄存器	PMC_PCK0	读 / 写	0x0
0x0044	可编程时钟 1 寄存器	PMC_PCK1	读 / 写	0x0
0x0048	可编程时钟 2 寄存器	PMC_PCK2	读 / 写	0x0
0x004C	可编程时钟 3 寄存器	PMC_PCK3	读 / 写	0x0
0x0050	保留	–	–	–
0x0054	保留	–	–	–
0x0058	保留	–	–	–
0x005C	保留	–	–	–
0x0060	中断使能寄存器	PMC_IER	只写	--
0x0064	中断禁用寄存器	PMC_IDR	只写	--
0x0068	状态寄存器	PMC_SR	只读	--
0x006C	中断屏蔽寄存器	PMC_IMR	只读	0x0



PMC 系统时钟使能寄存器

寄存器名称 : PMC_SCER

访问类型 : 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	PCK3	PCK2	PCK1	PCK0
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	–	UHP	–	MCKUDP	UDP	PCK

- **PCK: 处理器时钟使能**

0 = 无效。

1 = 使能处理器时钟。

- **UDP: USB 器件端口时钟使能**

0 = 无效。

1 = 使能 USB 器件端口 48 MHz 时钟。

- **MCKUDP: 挂起使能时 USB 器件端口主机时钟自动禁用**

0 = 无效。

1 = 当挂起条件出现时，使能 USB 器件端口主机时钟自动禁用。

- **UHP: USB 主机端口时钟使能**

0 = 无效。

1 = 使能 USB 主机端口的 48 MHz 时钟。

- **PCK0...PCK3: 可编程时钟输出使能**

0 = 无效。

1 = 使能相应的可编程时钟输出。

PMC 系统时钟禁用寄存器

寄存器名称 : PMC_SCDR

访问类型 : 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	PCK3	PCK2	PCK1	PCK0

7	6	5	4	3	2	1	0
–	–	–	UHP	–	MCKUDP	UDP	PCK

- **PCK: 处理器时钟禁用**

0 = 无效。

1 = 禁用处理器时钟，用来使处理器进入空闲模式。

- **UDP: USB 器件端口时钟禁用**

0 = 无效。

1 = 禁用 USB 器件端口 48 MHz 时钟。

- **MCKUDP: 挂起使能时 USB 器件端口主机时钟自动禁用**

0 = 无效。

1 = 当挂起条件出现时，禁用 USB 器件端口主机时钟自动禁用。

- **UHP: USB 主机端口时钟禁用**

0 = 无效。

1 = 禁用 USB 主机端口的 48 MHz 时钟。

- **PCK0...PCK3: 可编程时钟输出禁用**

0 = 无效。

1 = 禁用相应的可编程时钟输出。



PMC 系统时钟状态寄存器

寄存器名称 : PMC_SCSR

访问类型 : 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	PCK3	PCK2	PCK1	PCK0
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	–	UHP	–	MCKUDP	UDP	PCK

- **PCK: 处理器时钟状态**

0 = 处理器时钟禁用。

1 = 处理器时钟使能。

- **UDP: USB 器件端口状态**

0 = USB 器件端口 48 MHz 时钟禁用。

1 = USB 器件端口 48 MHz 时钟使能。

- **MCKUDP: 挂起使能时 USB 器件端口主机时钟自动禁用**

0 = 当挂起条件出现时 ， USB 器件端口主机时钟自动禁用被禁用。

1 = 当挂起条件出现时 ， USB 器件端口主机时钟自动禁用被使能。

- **UHP: USB 主机端口时钟状态**

0 = USB 主机端口的 48 MHz 时钟禁用。

1 = USB 主机端口的 48 MHz 时钟使能。

- **PCK0...PCK3: 可编程时钟输出状态**

0 = 相应的可编程时钟输出禁用。

1 = 相应的可编程时钟输出使能。

PMC 外设时钟使能寄存器

寄存器名称 : PMC_PCER

访问类型 : 只写

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PID7	PID6	PID5	PID4	PID3	PID2	–	–

- **PID2...PID31: 外设时钟使能**

0 = 无效。

1 = 使能相应外设时钟。

PMC 外设时钟禁用寄存器
寄存器名称 : PMC_PCDR
访问类型 : 只写

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	–	–

- **PID2...PID31: 外设时钟禁用**

0 = 无效。

1 = 禁用相应外设时钟。

PMC 外设时钟状态寄存器
寄存器名称 : PMC_PCSR
访问类型 : 只读

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	–	–

- **PID2...PID31: 外设时钟状态**

0 = 相应外设时钟禁用。

1 = 相应外设时钟使能。



PMC 时钟发生器主振荡器寄存器
寄存器名称： CKGR_MOR
访问类型： 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
OSCOUNT							

7	6	5	4	3	2	1	0
–	–	–	–	–	–	-	MOSCEN

- **MOSCEN: 主振荡器使能**

0 = 主振荡器禁用。主时钟信号与 XIN 连接。

1 = 主振荡器使能。晶体连接在 XIN 与 XOUT 间。

- **OSCOUNT: 主振荡器启动时间**

指定慢时钟周期数作为主振荡器启动时间。

PMC 时钟发生器主时钟频率寄存器
寄存器名称： CKGR_MCFR
访问类型： 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	MAINRDY

15	14	13	12	11	10	9	8
MAINF							

7	6	5	4	3	2	1	0
MAINF							

- **MAINF: 主时钟频率**

给出 16 个慢时钟周期中主时钟周期数。

- **MAINRDY: 主时钟就绪**

0 = MAINF 值无效或主振荡器禁用。

1 = 主振荡器提前使能且 MAINF 值有效。

PMC 时钟发生器 PLL A 寄存器

寄存器名称 : CKGR_PLLAR

访问类型 : 读 / 写

31	30	29	28	27	26	25	24
–	–	1	–	–	MULA		

23	22	21	20	19	18	17	16
MULA							

15	14	13	12	11	10	9	8
OUTA		PLLACOUNT					

7	6	5	4	3	2	1	0
DIVA							

在使用时钟发生器前需检查 PLL A 输入频率可能的限制及乘数。

- **DIVA: 分频器 A**

DIVA	分频器选择
0	分频器输出为 0
1	绕过分频器
2 - 255	分频器输出是主时钟由 DIVA 分频后的值

- **PLLACOUNT: PLL A 计数器**

给定 CKGR_PLLAR 写入后而 PMC_SR 中 LOCKA 位置位前的慢时钟周期数。

- **OUTA: PLL A 时钟频率范围**

OUTA		PLL A 频率输出范围
0	0	80 MHz 到 160 MHz
0	1	保留
1	0	150 MHz 到 240 MHz
1	1	保留

- **MULA: PLL A 乘法器**

0 = PLL A 无效。

1 到 2047 = PLL A 时钟频率为 PLL A 输入频率与 MULA+ 1 的乘积。





PMC 时钟发生器 PLL B 寄存器

寄存器名称： CKGR_PLLBR
访问类型： 读 / 写

31	30	29	28	27	26	25	24
–	–	–	USB_96M	–	MULB		

23	22	21	20	19	18	17	16
MULB							

15	14	13	12	11	10	9	8
OUTB		PLLBCOUNT					

7	6	5	4	3	2	1	0
DIVB							

- **DIVB: 分频器 B**

DIVB	分频器选择
0	分频器输出为 0
1	绕过分频器
2 - 255	分频器输出是所选时钟由 DIVB 分频后的值

- **PLLBCOUNT: PLL B 计数器**

给定 CKGR_PLLBR 写入后而 PMC_SR 中 LOCKB 位置位前的慢时钟周期数。

- **OUTB: PLL B 时钟频率范围**

OUTB		PLL B 时钟频率范围
0	0	80 MHz 到 160 MHz
0	1	保留
1	0	150 MHz 到 240 MHz
1	1	保留

- **MULB: PLL B 乘法器**

0 = PLL B 无效。

1 到 2047 = PLL B 时钟频率为 PLL B 输入频率与 MULB+ 1 的乘积。

- **USB_96M: 2 倍分频器使能 (仅在基于 ARM9 内核的系统中)**

0 = USB 端口时钟为 PLL B 时钟，因此 PMC 时钟发生器必须编程 PLL B 时钟为 48 MHz。

1 = USB 端口时钟为 PLL B 时钟 2 倍分频，因此 PMC 时钟发生器必须编程 PLL B 时钟为 96 MHz。

PMC 主机时钟寄存器

寄存器名称： PMC_MCKR

访问类型： 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	MDIV	
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–		PRES			CSS	

注意：写入 PMC_MCKR 的值不能与 PMC_MCKR 当前值相同。

- **CSS: 主机时钟选择**

CSS		时钟源选择
0	0	慢时钟
0	1	主时钟
1	0	PLL A 时钟
1	1	PLL B 时钟

- **PRES: 主机时钟预分频**

PRES			主机时钟
0	0	0	选定时钟
0	0	1	选定时钟除以 2
0	1	0	选定时钟除以 4
0	1	1	选定时钟除以 8
1	0	0	选定时钟除以 16
1	0	1	选定时钟除以 32
1	1	0	选定时钟除以 64
1	1	1	保留

- **MDIV: 主机时钟分频 (仅基于 ARM9 内核系统)**

0 = 主机时钟与处理器时钟相同。

1 = 处理器时钟是主机时钟的两倍。

2 = 处理器时钟是主机时钟的三倍。

3 = 处理器时钟是主机时钟的四倍。



PMC 可编程时钟寄存器 0 到 3

寄存器名称 : PMC_PCK0..PMC_PCK3

访问类型 : 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	PRES			CSS	

- **CSS: 主机时钟选择**

CSS		时钟源选择
0	0	慢时钟
0	1	主时钟
1	0	PLL A 时钟
1	1	PLL B 时钟

- **PRES: 可编程时钟预分频**

PRES			主机时钟
0	0	0	选定时钟
0	0	1	选定时钟除以 2
0	1	0	选定时钟除以 4
0	1	1	选定时钟除以 8
1	0	0	选定时钟除以 16
1	0	1	选定时钟除以 32
1	1	0	选定时钟除以 64
1	1	1	保留

PMC 中断使能寄存器

寄存器名称： PMC_IER

访问类型： 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	PCK3RDY	PCK2RDY	PCK1RDY	PCK0RDY
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	–	–	MCKRDY	LOCKB	LOCKA	MOSCS

- **MOSCS: 主振荡器状态**
- **LOCKA: PLL A 锁定**
- **LOCKB: PLL B 锁定**
- **MCKRDY: 主机时钟就绪**
- **PCK0RDY - PCK3RDY: 可编程时钟就绪**

0 = 无效。

1 = 使能相应中断。

PMC 中断禁用寄存器

寄存器名称： PMC_IDR

访问类型： 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	PCK3RDY	PCK2RDY	PCK1RDY	PCK0RDY
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	–	–	MCKRDY	LOCKB	LOCKA	MOSCS

- **MOSCS: 主振荡器状态**
- **LOCKA: PLL A 锁定**
- **LOCKB: PLL B 锁定**
- **MCKRDY: 主机时钟就绪**
- **PCK0RDY - PCK3RDY: 可编程时钟就绪**

0 = 无效。

1 = 禁用相应中断。



PMC 状态寄存器

寄存器名称： PMC_SR

访问类型： 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	PCK3RDY	PCK2RDY	PCK1RDY	PCK0RDY
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	–	–	MCKRDY	LOCKB	LOCKA	MOSCS

- **MOSCS: MOSCS 标志状态**

0 = 主振荡器不稳定。

1 = 主振荡器稳定。

- **LOCKA: PLLA 锁定状态**

0 = PLLL A 未锁定。

1 = PLL A 锁定。

- **LOCKB: PLLB 锁定状态**

0 = PLL B 未锁定。

1 = PLL B 锁定。

- **MCKRDY: 主机时钟状态**

0 = 主机时钟未就绪。

1 = 主机时钟就绪。

- **PCK0RDY - PCK3RDY: 可编程时钟就绪状态**

0 = 可编程时钟 0 到 3 未就绪。

1 = 可编程时钟 0 到 3 就绪。

PMC 中断屏蔽寄存器

寄存器名称 : PMC_IMR

访问类型 : 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	PCK3RDY	PCK2RDY	PCK1RDY	PCK0RDY
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	–	–	MCKRDY	LOCKB	LOCKA	MOSCS

- **MOSCS: 主振荡器状态**
- **LOCKA: PLL A 锁定**
- **LOCKB: PLL B 锁定**
- **MCKRDY: 主机时钟就绪**
- **PCK0RDY - PCK3RDY: 可编程时钟就绪**
- **MOSCS: MOSCS 中断屏蔽**

0 = 相应中断使能。

1 = 相应中断禁用。

系统定时器 (ST)

概述

系统定时器 (ST) 模块集成了三个不同的自由运行的定时器：

- 周期间隔定时器 (PIT) 为操作系统设置时基。
- 看门狗定时器 (WDT)，软件死锁时可复位系统。
- 实时定时器 (RTT)，计数消逝的时间。

这些定时器使用电源管理控制器提供的慢时钟计数。典型值该时钟频率为 32.768 kHz，但系统定时器可能要配置支持另一频率。

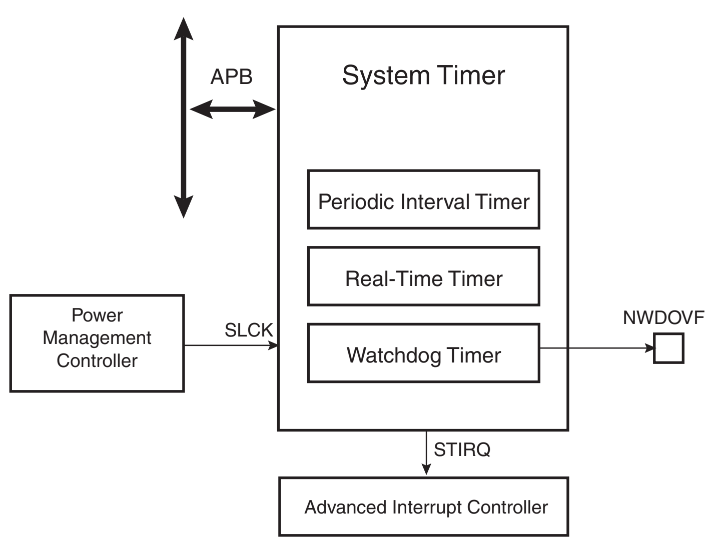
系统定时器提供了一条与高级中断控制器 (AIC) 连接的中断线。中断处理请求在配置系统定时器前编程 AIC，系统定时器中断线与第一个中断线连接并共享调试单元 (DBGU) 入口及实时时钟 (RTC)。当该共享中断源 1 被触发后，需要程序员决定中断源。

系统定时器重要特性：

- 一个周期间隔定时器， 16 位可编程计数器
- 一个看门狗定时器， 16 位可编程计数器
- 一个实时定时器， 20 位自由运行计数器
- 事件中断发生器

方框图

Figure 129. 系统定时器框图



应用框图

Figure 130. 应用框图

OS or RTOS Scheduler	Date, Time and Alarm Manager	System Survey Manager
PIT	RTT	WDT

³³¾þÙðŽý²

电源管理

系统定时器时钟为连续的 32768 Hz 时钟。电源管理控制器对系统定时器工作无效。

中断源

系统定时器中断通常与高级中断控制器源 1 连接。该中断线是系统外设中断线 (系统定时器、实时时钟、电源管理控制器、存储控制器) 线或的结果。当系统中断发生，服务程序必须首先确定中断的起因。这是通过正确读取上面涉及到的系统外设的状态寄存器来完成的。

看门狗溢出

系统定时器可驱动 NWDOVF 引脚。该引脚可由产品执行。当它执行时，该引脚可与 PIO 控制器复用，尽管推荐使用专用引脚实现看门狗功能 。若 NWDOVF 复用到 PIO 控制器上，后者须在使用该引脚为 NWDOVF 前，将其分配为看门狗功能。

当未执行时，对相关位及寄存器编程对系统定时器行为无影响。

功能说明

系统定时器时钟

系统定时器仅使用 SLCK 时钟，这样即使在电源管理控制器编程将芯片置于慢时钟模式下，系统定时器仍能提供周期性的、看门狗、二次变化或报警中断。若产品能返回慢时钟振荡器，则系统定时器可继续运行。

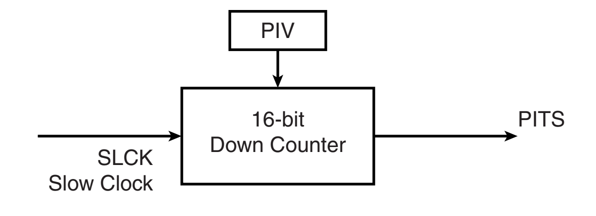
周期间隔定时器 (PIT)

周期间隔定时器可用来向操作系统提供周期性的中断。 PIT 的复位值为 0。它基于一个 16 位的递减计数器，计数器预先载入 ST_PIMR (周期间隔模式寄存器) 中的编程值。当 PIT 计数器达到 0， ST_SR (状态寄存器) 中的 PITS 位置位，若中断使能的话，将产生一个中断。

然后计数器自动重载并重新启动。任何时间对 ST_PIMR 的写操作将立即从新编程值开始重载并重新启动递减计数器。

警告：若 ST_PIMR 的编程周期低于或等于当前 MCK 周期，更新 PITS 状态位，其相关中断将不可预测。

Figure 131. 周期间隔定时器



看门狗定时器 (WDT)

看门狗定时器可用来软件陷入死锁时系统锁定。其结构基于一个 16 位递减计数器，计数器值从 ST_WDMR (看门狗模式寄存器) 中载入。

复位时， ST_WDMR 值为 0x00020000，对应于计数器最大值。在 8 个慢时钟周期中看门狗溢出 (ST_WDMR 寄存器中 EXTEN 位置位) 时，看门狗溢出信号为低。

使用慢时钟信号的128倍分频信号可确定最大看门狗周期为256秒(在典型慢时钟频率为32.768 kHz 的情况下)。

正常操作下，定时器溢出发生前，用户通过设置 ST_CR (控制寄存器) 的 WDRST 位，定期重载看门狗值。

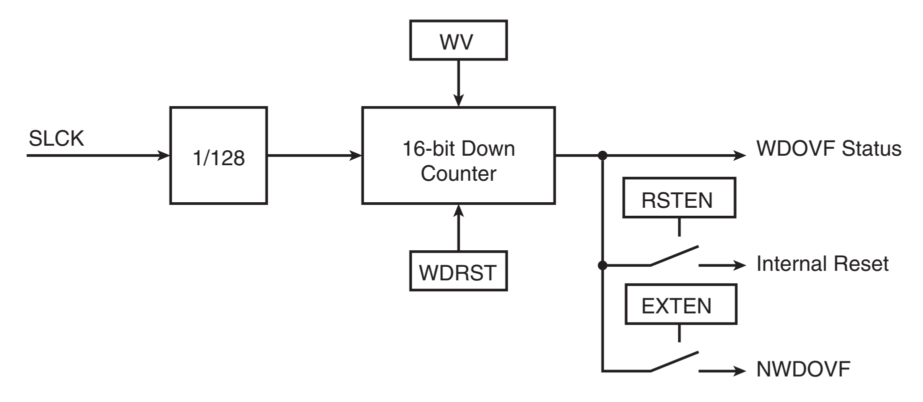
若溢出出现，看门狗定时器：

- 设置中断产生的 ST_SR (状态寄存器) 的 WDOVF 位。

- 若 ST_WDMR 中的 EXTEN 位置位，在外部看门狗溢出信号中产生一个 8 周期的慢时钟脉冲。
- 若 ST_WDMR 中 RSTEN 置位，产生一个中断复位。
- 重载并重启递减计数器。

写 ST_WDMR 不会重载或重启递减计数器。当对 ST_CR 写入后，看门狗计数器立即由 ST_WDMR 中载入值并重启，慢时钟的 128 分频器也立即复位并重启。

Figure 132. 看门狗定时器



实时定时器 (RTT)

实时定时器用来计算消逝的时间。它是基于一个 20 位的计数器，其值由慢时钟被可编程值分频后的值提供。复位时，该值设为 0x8000，对应当慢时钟为 32.768 Hz 时，相应实时计数器频率为 1 Hz。 20 位计数器可向上计数到 1048576 秒，即超过 12 天，然后返回 0。

可在任意时刻从 ST_CRTR (当前实时寄存器) 中读取实时定时器值。由于该值可与主时钟异步更新，所以推荐在相同值时对该寄存器读两次，以增加返回值的精度。

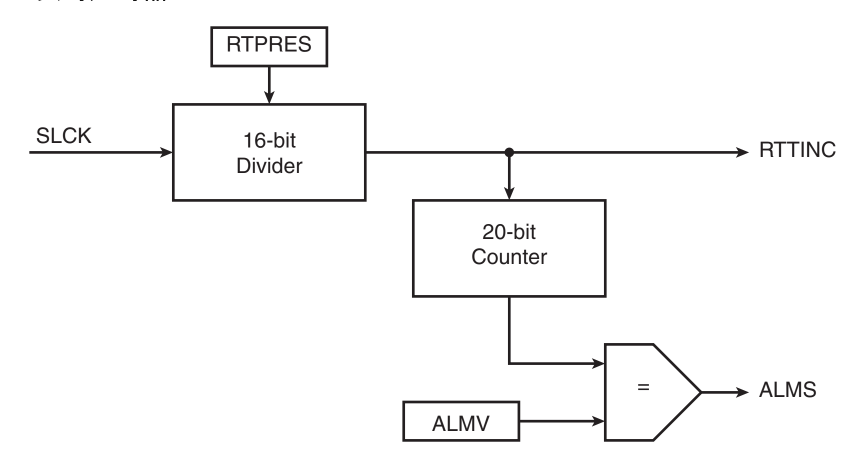
计数器中当前值与写入报警寄存器 ST_RTAR (实时报警寄存器) 的值进行比较。若计数器值达到报警值，设置 TC_SR 中的 ALMS 位。复位后，报警寄存器设置为最大值，相应值为 0。

20位计数器每次增加时，设置ST_SR中的RTTINC位，该位可用来启动中断，或产生一个1秒信号。

写 ST_RTMR 将立即对时钟分频器载入新值并重启，这同样会导致 20 位计数器复位。

警告：若 RTPRES 的编程设置周期值小于或等于当前 MCK 周期值， RTTINC 与 ALMS 状态位更新及它们的相关中断结果无法预料。

Figure 133. 实时定时器



系统定时器 (ST) 用户接口

Table 62. 系统定时器寄存器

偏移	寄存器	名称	访问类型	复位值
0x0000	控制寄存器	ST_CR	只写	–
0x0004	周期间隔模式寄存器	ST_PIMR	读 / 写	0x00000000
0x0008	看门狗模式寄存器	ST_WDMR	读 / 写	0x00020000
0x000C	实时模式寄存器	ST_RTMR	读 / 写	0x00008000
0x0010	状态寄存器	ST_SR	只读	–
0x0014	中断使能寄存器	ST_IER	只写	–
0x0018	中断禁用寄存器	ST_IDR	只写	–
0x001C	中断屏蔽寄存器	ST_IMR	只写	0x0
0x0020	实时报警寄存器	ST_RTAR	读 / 写	0x0
0x0024	当前实时寄存器	ST_CRTR	只读	0x0

ST 控制寄存器

寄存器名称 : ST_CR

访问类型 : 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	–	–	–	–	–	WDRST

- **WDRST: 看门狗定时器重启**

0 = 无效。

1 = 重载看门狗定时器启动值。

ST 周期间隔模式寄存器

寄存器名称： ST_PIMR

访问类型： 读 / 写

–	–	–	–	–	–	–	–

–	–	–	–	–	–	–	–

PIV							

PIV							

- **PIV: 周期间隔值**

定义了加载在周期间隔定时器 16 位计数器中的值。最大周期可通过编程 PIV 为 0x0，对应 65536 个慢时钟周期。

ST 看门狗模式寄存器

寄存器名称： ST_WDMR

访问类型： 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	EXTEN	RSTEN

15	14	13	12	11	10	9	8
WDV							

7	6	5	4	3	2	1	0
WDV							

- **WDV: 看门狗计数器值**

定义载入 16 位计数器的值。最大周期可通过编程 WDV 为 0x0，对应 65536x 128 个慢时钟周期。

- **RSTEN: 复位使能**

0 = 当看门狗溢出发生时，不产生复位。

1 = 当看门狗溢出发生时产生一个内部复位。

- **EXTEN: 外部信号判决使能**

0 = 当看门狗溢出发生时，看门狗溢出不会置低。

1 = 当看门狗溢出发生时，看门狗溢出在 8 周期慢时钟中置低。

ST 实时模式寄存器

寄存器名称： ST_RTMR

访问类型： 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
RTPRES							

7	6	5	4	3	2	1	0
RTPRES							

- **RTPRES: 实时定时器预分频值**

定义实时定时器递增所需的 SLCK 周期数。最大周期可通过编程 RTPRES 为 0x0，对应 65536 个慢时钟周期。

ST 状态寄存器

寄存器名称： ST_SR

访问类型： 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	ALMS	RTTINC	WDOVF	PITS

- **PITS: 周期间隔定时器状态**

0 = 上次读状态寄存器后，周期间隔定时器未达到 0。

1 = 上次读状态寄存器后，周期间隔定时器已达到 0。

- **WDOVF: 看门狗溢出**

0 = 上次读状态寄存器后，看门狗定时器未达到 0。

1 = 上次读状态寄存器后，看门狗定时器已达到 0。

- **RTTINC: 实时定时器递增**

0 = 上次读状态寄存器后，实时定时器未增加。

1 = 上次读状态寄存器后，实时定时器已增加。

- **ALMS: 报警状态**

0 = 上次读状态寄存器后，未检测到报警比较。

1 = 上次读状态寄存器后，已检测到报警比较。



ST 中断使能寄存器

寄存器名称： ST_IER

访问类型： 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	ALMS	RTTINC	WDOVF	PITS

- **PITS: 周期间隔定时器状态中断使能**
- **WDOVF: 看门狗溢出中断使能**
- **RTTINC: 实时定时器递增中断使能**
- **ALMS: 报警状态中断使能**

0 = 无效。

1 = 使能相应中断。

ST 中断禁用寄存器

寄存器名称： ST_IDR

访问类型： 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	ALMS	RTTINC	WDOVF	PITS

- **PITS: 周期间隔定时器状态中断禁用**
- **WDOVF: 看门狗溢出中断禁用**
- **RTTINC: 实时定时器递增中断禁用**
- **ALMS: 报警状态中断禁用**

0 = 无效。

1 = 禁用相应中断。

ST 中断屏蔽寄存器

寄存器名称 : ST_IMR

访问类型 : 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	–	–	ALMS	RTTINC	WDOVF	PITS

- **PITS: 周期间隔定时器状态中断屏蔽**
- **WDOVF: 看门狗溢出中断屏蔽**
- **RTTINC: 实时定时器递增中断屏蔽**
- **ALMS: 报警状态中断屏蔽**

0 = 相应中断禁用。

1 = 相应中断使能。

ST 实时报警寄存器

寄存器名称 : ST_RTAR

访问类型 : 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	ALMV			
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
ALMV							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ALMV							

- **ALMV: 报警值**

定义与实时定时器比较的报警值。ALMS 位激活前最大延迟可编程 ALMV 为 0x0，对应 1048576 秒。

ST 当前实时寄存器

寄存器名称 : ST_CRTR

访问类型 : 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	CRTV			

15	14	13	12	11	10	9	8
CRTV							

7	6	5	4	3	2	1	0
CRTV							

- **CRTV: 当前实时值**

返回实时定时器当前值。

实时控制器 (RTC)

概述

实时时钟 (RTC) 外设专为低功耗要求设计。

它包含一个完整的带有报警的日时间时钟及一个 200 年的日历，及可编程周期中断。报警与日历寄存器可通过 32 位数据总线访问。

时间与日历值可编码为二进制编码的十进制(BCD)格式。时间格式可为24小时模式或有AM/PM 指示器的 12 小时模式。

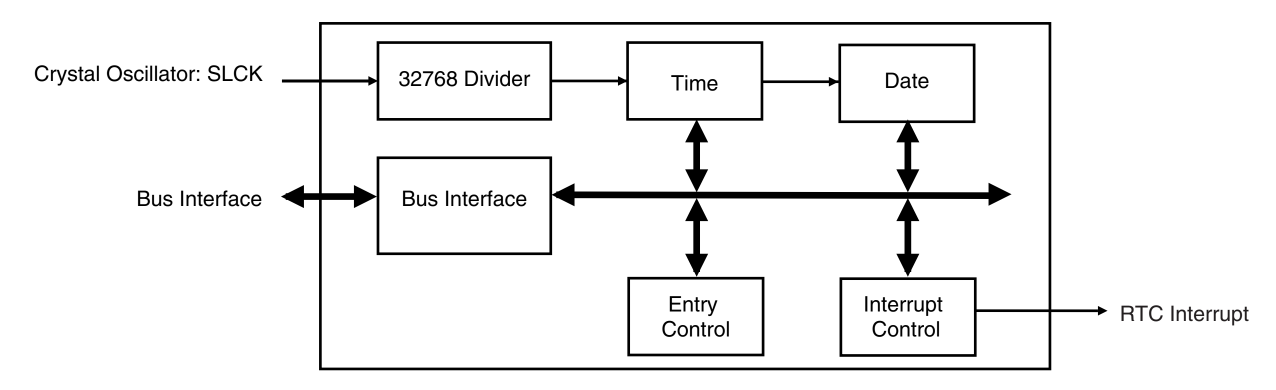
更新时间和日历域及配置报警域可通过 32 位数据总线上的一个并行捕获执行。执行入口控制以避免加载到一个与当前 month/year/century 或与 BCD 格式不兼容的数据。

RTC 重要特性包括：

- 低功耗
- 全异步设计
- 200 年日历
- 可编程周期性中断
- 报警与更新并行载入
- 报警控制与时间 / 日历数据更新

方框图

Figure 134. RTC 框图



附属产品

电源管理

实时时钟为连续的 32768 Hz 时钟。电源管理控制器对 RTC 工作无效。

中断

RTC 中断与高级中断控制器的中断源 1 (IRQ1) 连接。该中断线是系统外设中断线 (系统定时器、实时时钟、电源管理控制器、存储控制器) 线或的结果。当系统中断发生，服务程序必须首先确定中断的起因。这是通过正确读取上面涉及到的系统外设的状态寄存器来完成的。

功能说明

RTC 提供了一个二进制编码的十进制 (BCD) 时钟，包括世纪 (19/20)、年 (有闰年)、月、日期、日、时、分与秒。

有效的年范围为 1900 到 2099， 200 年的日历可适用 Y2K。

RTC 可工作在 24 小时模式下或有 AM/PM 指示器的 12 小时模式。



包括了闰年的修正 (所有可被 4 整除的是闰年，包括 2000 年)。修正到达 2099。

硬件复位后，日历值为 1998 年， 1 月 1 日，星期四。

参考时钟

参考时钟为慢时钟 (SLCK)。它可由 Atmel OSC55 或 OSC56 单元 (或等效单元) 及外部的 32.768 kHz 晶振驱动。

处理器低功耗模式下 (空闲模式)，振荡器运行和功耗均很严格。晶振的选择需要考虑当前功耗及由于温度对电路影响发生的频率漂移而影响精度。

定时

普通模式下， RTC 实时以秒更新内部的秒计数器，以分更新内部分计数器，等等。

由于 RTC 的异步操作对芯片复位下影响，为确定从 RTC 寄存器所读取的值有效且稳定，必须对这些寄存器读取两次。若两次值相同，则该值有效。因此最少需要访问两次，最多需要访问三次。

报警

RTC 有 5 个可编程域：月、日期、时、分、秒。

为报警需要，各个域均可被使能或禁用：

- 若所有域使能，在给定的月、日期时、分、秒产生报警标志 (相应标志位出现，若使能时产生中断)。
- 若仅有 “ 秒 ” 域使能，则每分钟均产生一个报警。

根据使能域的组合，报警时间可从分到 365/366 天。

错误检查

在访问世纪、年、月、日期、日、时、分、秒及报警时要确认用户接口数据。对于非法的 BCD 入口访问如非法的将月数据当成年或世纪数据配置等进行检查。

若时间域中有错误，数据不会载入寄存器 / 计数器，并在有效寄存器中设置标志。用户不能复位该标志。该标志在编程设置了一个可接收的值后复位。这可避免任何对硬件单方面的影响。对于报警的处理步骤类似。

执行下列检查：

1. 世纪 (检查是否在 19 - 20 间)
2. 年 (BCD 入口检查)
3. 日期 (检查是否在 01 - 31 间)
4. 月 (检查是否在 01 - 12 间，检查 “ 日期 ” 是否合法)
5. Day (检查是否在 1 - 7 间)
6. 时 (BCD 检查 : 24 小时模式，检查是否在 00 - 23 间，及检查 AM/PM 标志是否未置位； 12 小时模式，检查是否在 01 - 12 间)
7. 分 (检查 BCD 及是否在 00 - 59 间)
8. 秒 (检查 BCD 及是否在 00 - 59 间)

Note: 若通过 RTC_MODE 寄存器选择 12 小时模式，可对 12 小时值编程，RTC_TIME 中返回值为对应的 24 小时值。入口控制检查 AM/PM 指示器值 (RTC_TIME 寄存器的 22 位) 以确定检查范围。

更新时间 / 日历

为更新时间 / 日历域中的值，必须通过在控制寄存器的相应位设置停止 RTC。 UPDTIM 位设置以更新时间域 (时、分、秒)， UPDCAL 位设置以更新日历域 (世纪、年、月、日期、日)。

然后须等待轮询或状态寄存器的 ACKUPD 位中断 (若使能)。一旦该位为 1，可对适当寄存器写入。

一旦更新结束，必须对控制寄存器的 UPDTIM 与 / 或 UPDCAL 位复位 (0)。

对日历域编程时，时间域编程使能。以避免日历更新阶段时的时间丢失。在连续更新操作中，要在对 RTC_CR(控制寄存器) 的 UPDTIM/UPDCAL 位重设置后，对这些位设置前，等待至少 1

秒。通过在设置 UPDTIM/UPDCAL 位前等待状态寄存器的 SEC 标志来实现。对 UPDTIM/UPDCAL 重新设置后， SEC 标志也将清除。



实时控制器 (RTC) 用户接口

Table 63. RTC 寄存器映射

偏移	寄存器	寄存器名称	访问类型	复位值
0x00	RTC 控制寄存器	RTC_CR	读 / 写	0x0
0x04	RTC 模式寄存器	RTC_MR	读 / 写	0x0
0x08	RTC 时间寄存器	RTC_TIMR	读 / 写	0x0
0x0C	RTC 日历寄存器	RTC_CALR	读 / 写	0x01819819
0x10	RTC 时间报警寄存器	RTC_TIMALR	读 / 写	0x0
0x14	RTC 日历报警寄存器	RTC_CALALR	读 / 写	0x01010000
0x18	RTC 状态寄存器	RTC_SR	只读	0x0
0x1C	RTC 状态清除命令寄存器	RTC_SCCR	只写	---
0x20	RTC 中断使能寄存器	RTC_IER	只写	---
0x24	RTC 中断禁用寄存器	RTC_IDR	只写	---
0x28	RTC 中断屏蔽寄存器	RTC_IMR	只读	0x0
0x2C	RTC 有效入口寄存器	RTC_VER	只读	0x0

RTC 控制寄存器

寄存器名称 : RTC_CR

访问类型 : 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	CALEVSEL	

15	14	13	12	11	10	9	8
–	–	–	–	–	–	TIMEVSEL	

7	6	5	4	3	2	1	0
–	–	–	–	–	–	UPDCAL	UPDTIM

- **UPDTIM: 更新请求时间寄存器**

0 = 无效。

1 = 停止 RTC 时间计数。

时间计数包括秒、分、时计数器。时间计数器可在该位设置时编程进行编程操作，并由状态寄存器的 ACKUPD 位应答。

- **UPDCAL: 更新请求日历寄存器**

0 = 无效。

1 = 停止 RTC 日历计数。

日历计数包括日。日期、月、年及世纪计数器。一旦该位置位，可对日历计数器编程。

- **TIMEVSEL: 时间事件选择**

根据 TIMEVSEL 值，在 RTC_SR (状态寄存器) 的 TIMEV 位产生标志。

0 = 分改变。

1 = 时改变。

2 = 每天的午夜。

3 = 每天的中午。

- **CALEVSEL: 日历事件选择**

根据 CALEVSEL 值，在 RTC_SR 的 CALEV 位产生标志。

0 = 周改变 (每个周一的 00:00:00)。

1 = 月改变 (每月 1 日的 00:00:00)。

2, 3 = 年改变 (每个 1 月 1 日的 00:00:00)。



RTC 模式寄存器

寄存器名称： RTC_MR

访问类型： 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	HRMOD

- **HRMOD: 12-/24- 小时模式**

0 = 选择 24 小时模式。

1 = 选择 12 小时模式。

所有未定义位读为 0。

RTC 时间寄存器

寄存器名称 : RTC_TIMR

访问类型 : 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	AMPM	HOUR					

15	14	13	12	11	10	9	8
–	MIN						

7	6	5	4	3	2	1	0
–	SEC						

- **SEC: 当前秒**

范围设置在 0 - 59 间 (BCD)。

最低 4 位编码，高位编码为十进制。

- **MIN: 当前分**

范围设置在 0 - 59 间 (BCD)。

最低 4 位编码，高位编码为十进制。

- **HOUR: 当前时**

12 小时模式下，范围设置在 1 - 12 间 (BCD) ； 24 小时模式下，范围设置在 0 - 23 间 (BCD)。

- **AMPM: AM/PM 指示器**

该位是 12 小时模式下 AM/PM 指示位。

0 = AM。

1 = PM。

所有未定义位读为 0。





RTC 日历寄存器

寄存器名称： RTC_CALR

访问类型： 读 / 写

31	30	29	28	27	26	25	24
–	–	DATE					
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
DAY			MONTH				
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
YEAR							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	CENT						

- **CENT: 当前世纪**

范围设置在 19 - 20 间 (BCD)。

最低 4 位编码，高位编码为十进制。

- **YEAR: 当前年**

范围设置在 00 - 99 间 (BCD)。

最低 4 位编码，高位编码为十进制。

- **MONTH: 当前月**

范围设置在 01 - 12 间 (BCD)。

最低 4 位编码，高位编码为十进制。

- **DAY: 当前天**

范围设置在 1 - 7 (BCD).

编码序号 (哪个序号代表哪天) 由用户定义，它对日期计数器无影响。

- **DATE: 当前日期**

范围设置在 01 - 31 间 (BCD)。

最低 4 位编码，高位编码为十进制。

所有未定义位读为 0。

RTC 时间报警寄存器

寄存器名称： RTC_TIMALR

访问类型： 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
HOUREN	AMPM	HOUR					

15	14	13	12	11	10	9	8
MINEN	MIN						

7	6	5	4	3	2	1	0
SECEN	SEC						

- **SEC: 秒报警**

该域与 BCD 编码的秒计数器相关。

- **SECEN: 秒报警使能**

0 = 秒匹配报警禁用。

1 = 秒匹配报警使能。

- **MIN: 分报警**

该域与 BCD 编码的分计数器相关。

- **MINEN: 分报警使能**

0 = 分匹配报警禁用。

1 = 分匹配报警使能。

- **HOUR: 时报警**

该域与 BCD 编码的时计数器相关。

- **AMPM: AM/PM 指示器**

该域与 BCD 编码的时计数器相关。

- **HOUREN: 时报警使能**

0 = 时匹配报警禁用。

1 = 时匹配报警使能。

RTC 日历报警寄存器

寄存器名称： RTC_CALALR

访问类型： 读 / 写

31	30	29	28	27	26	25	24
DATEEN	–	DATE					

23	22	21	20	19	18	17	16
MTHEN	–	–	MONTH				

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

- **MONTH: 月报警**

该域与 BCD 编码的月计数器相关。

- **MTHEN: 月报警使能**

0 = 月匹配报警禁用。

1 = 月匹配报警使能。

- **DATE: 日期报警**

该域与 BCD 编码的日期计数器相关。

- **DATEEN: 日期报警使能**

0 = 日期匹配报警禁用。

1 = 日期匹配报警使能。

RTC 状态寄存器

寄存器名称： RTC_SR

访问类型： 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	–	CALEV	TIMEV	SEC	ALARM	ACKUPD

- **ACKUPD: 更新应答**

0 = 时间与日历寄存器不能更新。

1 = 时间与日历寄存器可更新。

- **ALARM: 报警标志**

0 = 无报警匹配状态出现。

1 = 报警匹配状态出现。

- **SEC: 秒事件**

0 = 上次清零后无秒事件出现。

1 = 上次清零后至少有一次秒事件出现。

- **TIMEV: 时间事件**

0 = 上次清零后无时间事件出现。

1 = 上次清零后至少有一次时间事件出现。

时间事件在 RTC_CTRL (控制寄存器) 的 TIMEVSEL 域进行选择，可为下列任一事件：分改变、时改变、中午、午夜 (日改变)。

- **CALEV: 日历事件**

0 = 上次清零后无日历事件出现。

1 = 上次清零后至少有一次日历事件出现。

日历事件在 RTC_CTRL 的 CALEVSEL 域进行选择，可为下列任一事件：周改变、月改变、年改变。

RTC 状态清除命令寄存器

寄存器名称 : RTC_SCCR

访问类型 : 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	CALCLR	TIMCLR	SECCLR	ALRCLR	ACKCLR

- **状态标志清除**

0 = 无效。

1 = 清除状态寄存器 (RTC_SR) 的相应状态标志。

RTC 中断使能寄存器

寄存器名称 : RTC_IER

访问类型 : 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	CALEN	TIMEN	SECEN	ALREN	ACKEN

- **ACKEN: 应答更新中断使能**

0 = 无效。

1 = 更新中断应答使能。

- **ALREN: 报警中断使能**

0 = 无效。

1 = 报警中断使能。

- **SECEN: 秒事件中断使能**

0 = 无效。

1 = 秒周期中断使能。

- **TIMEN: 时间事件中断使能**

0 = 无效。

1 = 选定的时间事件中断使能。

- **CALEN: 日历事件中断使能**

0 = 无效。

1 = 选定的日历事件中断使能。



RTC 中断禁用使能

寄存器名称 : RTC_IDR

访问类型 : 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	CALDIS	TIMDIS	SECDIS	ALRDIS	ACKDIS

- **ACKDIS: 应答更新中断禁用**

0 = 无效。

1 = 更新中断应答禁用。

- **ALRDIS: 报警中断禁用**

0 = 无效。

1 = 报警中断禁用。

- **SECDIS: 秒事件中断禁用**

0 = 无效。

1 = 秒周期中断禁用。

- **TIMDIS: 时间事件中断禁用**

0 = 无效。

1 = 选定的时间事件中断禁用。

- **CALDIS: 日历事件中断禁用**

0 = 无效。

1 = 选定的日历事件中断禁用。

RTC 中断屏蔽寄存器

寄存器名称： RTC_IMR

访问类型： 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	CAL	TIM	SEC	ALR	ACK

- **ACK: 应答更新中断屏蔽**

0 = 更新中断应答禁用。

1 = 更新中断应答使能。

- **ALR: 报警中断屏蔽**

0 = 报警中断禁用。

1 = 报警中断使能。

- **SEC: 秒事件中断屏蔽**

0 = 秒周期中断禁用。

1 = 秒周期中断使能。

- **TIM: 时间事件中断屏蔽**

0 = 选定的时间事件中断禁用。

1 = 选定的时间事件中断使能。

- **CAL: 日历事件中断屏蔽**

0 = 选定的日历事件中断禁用。

1 = 选定的日历事件中断使能。

RTC 有效入口寄存器

寄存器名称： RTC_VER

访问类型： 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	–	–	NVCALAR	NVTIMALR	NVCAL	NVTIM

- **NVTIM: 无有效时间**

0 = RTC_TIMR (时间寄存器) 中未检测到有效数据。

1 = 上次编程设置后， RTC_TIMR 包含有效数据。

- **NVCAL: 无有效日历**

0 = RTC_CALR (日历寄存器) 中未检测到有效数据。

1 = 上次编程设置后， RTC_CALR 包含有效数据。

- **NVTIMALR: 无有效时间报警**

0 = RTC_TIMALR (时间报警寄存器) 中未检测到有效数据。

1 = 上次编程设置后，RTC_TIMALR 包含有效数据。

- **NVCALALR: 无有效日历报警**

0 = RTC_CALALR (日历报警寄存器) 中未检测到有效数据。

1 = 上次编程设置后， RTC_CALALR 包含有效数据。

调试单元 (DBGU)

概述

调试单元为处理器访问基于 Atmel 的 ARM 内核系统所有调试功能提供了一个单入口点。

调试单元的 2 引脚 UART 可用于多种调试与追踪目的，它为 in-situ 编程方案及调试监控通信提供了理想媒介。此外，与两个外设数据控制器连接的通道允许进行批处理以降低处理器时间。

调试单元也可使得由 ARM 处理器板上仿真器提供的调试通信通道 (DCC) 信号对软件可见。这些信号指示 DCC 读与写寄存器状态并产生一个 ARM 处理器中断，以使对 DCC 处理在中断控制下进行。

芯片标识符寄存器用来辨别器件及其版本。这些寄存器中有片上存储器大小与类型，及内置外设。

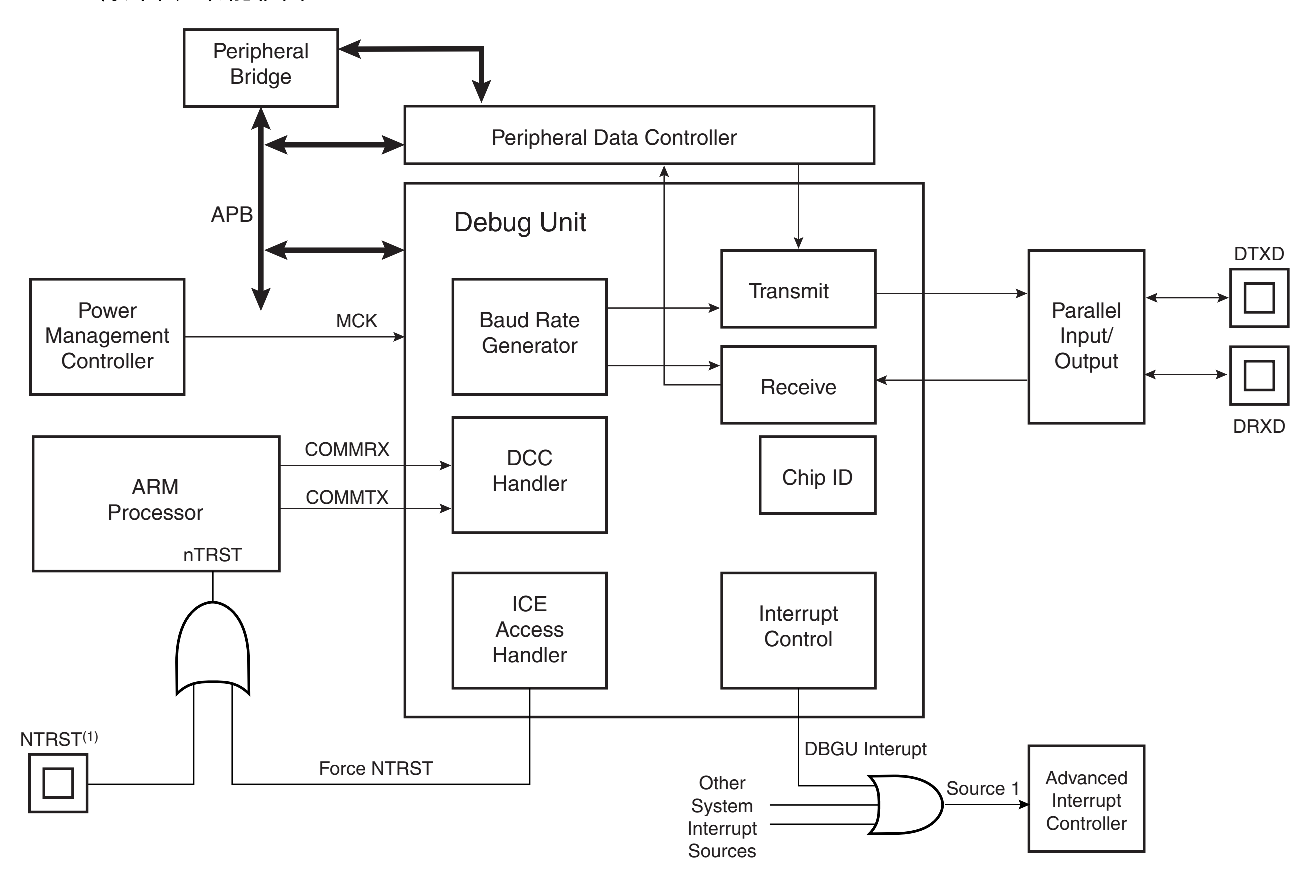
最后，调试单元还有强制 NTRST 功能，使得软件可决定是否阻止通过板上仿真器访问系统。这允许将保护代码存于 ROM 中。

调试单元重要特性如下：

- 为方便 Atmel 的基于 ARM 内核调试的系统外设
- 4 项功能
  - 2 引脚 UART
  - 支持调试通信信道 (DCC)
  - 芯片 ID 寄存器
  - ICE 访问限制
- 2 引脚 UART
  - 实现特性与标准 Atmel USART100% 兼容
  - 有通用可编程波特率发生器的独立接收器与发送器
  - 奇、偶、标志或空校验
  - 奇偶校验、帧及溢出误差检测
  - 自动回应、本地回环及远程回环通道模式
  - 中断发生
  - 支持与接收器与发送器的两 PDC 通道
- 调试特性通道支持
  - ARM 处理器提供了可见的 COMMRX 与 COMMTX 信号
  - 中断发生
- 芯片 ID 寄存器
  - 辨别器件版本、内置存储器大小及外设
- ICE 访问限制
  - 软件使能以防止通过 ARM 处理器的 ICE 访问系统
  - 在 ARM 处理器的 ICE 上插入 NTRST 线来防止访问

方框图

Figure 135. 调试单元功能框图

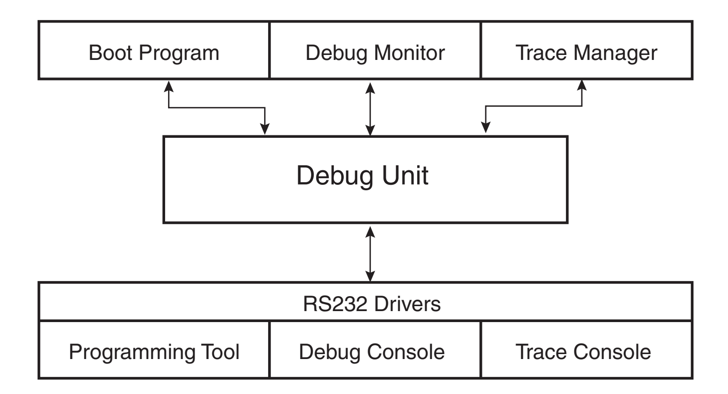


Note: 1. 若 NTRST 焊盘未完全绑定，它与 NRST 连接。

Table 64. 调试单元引脚说明

引脚名称	说明	类型
DRXD	调试接收数据	输入
DTXD	调试发送数据	输出

Figure 136. 调试单元应用示例



附属产品

I/O 线

根据产品集成，调试单元引脚可复用为 PIO 线。此时，程序员必须首先配置相应的 PIO 控制器以使能调试单元的 I/O 线操作。

电源管理

根据产品集成，调试单元时钟可由电源管理控制器来控制。此时，程序员必须首先配置 PMC 以使能调试单元时钟。通常此时的外设标识符为 1。

中断源

根据产品集成，调试单元中断线与高级中断控制器中的一个中断源连接。在配置调试单元前中断处理请求对 AIC 编程。通常，调试单元中断线与 AIC 中断源 1 连接，可共享实时时钟、系统定时器中断线及其它系统外设中断，见 Figure 135。为此程序员在触发源 1 时确定中断源。

UART 操作

调试单元作为 UART 运行时 (仅在异步模式下) 只支持 8 位字符处理 (有奇偶校验位)。没有时钟引脚。

调试单元的 UART 由独立工作的接收器与发送器及一个通用波特率发生器组成。计数器超时与发送器时间保护不执行。但是，所有执行特性与标准 USART 兼容。

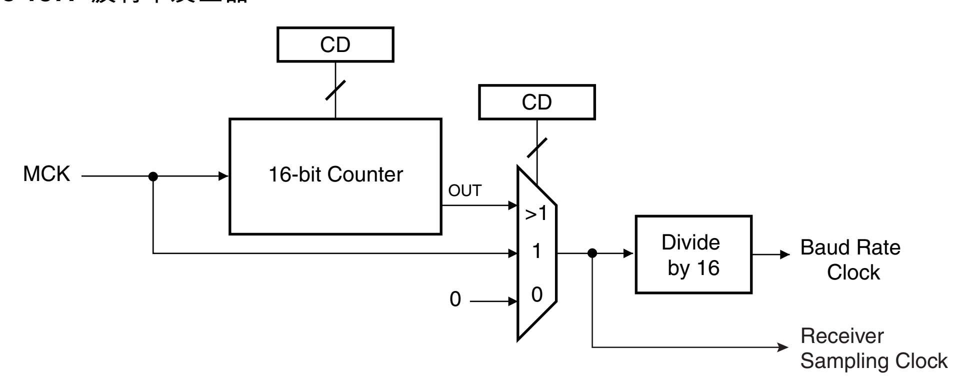
波特率发生器

波特率发生器向发送器与接收器提供名为波特率时钟的位周期时钟。

波特率时钟为主机时钟与 16 倍 DBGU_BRGR(波特率发生器寄存器)CD 值相除后的结果。若 DBGU_BRGR置为0 ,波特率时钟禁用且调试单元的UART无效。最大允许的波特率为主机时钟的 16 分频。最小允许的波特率为主机时钟的 16 x 65536 分频。

$$\text{Baud Rate} = \frac{MCK}{16 \times CD}$$

Figure 137. 波特率发生器



接收器

接收器复位，使能与禁用

器件复位后 ,调试单元接收器禁用 ,在使用前必须使能。接收器可通过在控制寄存器 DBGU_CR 的 RXEN 位写入 1 使能。该命令后，接收器开始寻找起始位。

程序员可在 DBGU_CR 寄存器的 RXDIS 写 1 来禁用接收器。若接收器正等待起始位，则立即停止。但若接收器已检测到起始位并正在接收数据，则将等待停止位出现后再停止。

程序员也可写 DBGU_CR 寄存器 RSTRX 位为 1 将接收器置于复位状态。此时，接收器立即停止当前操作并将其禁用。若当处理数据时设置 RSTRX，数据将丢失。

开始检测与数据采样

调试单元只支持异步操作，且仅影响接收器。调试单元接收器通过对 DRXD 信号采样来检测接收字符起始位。若采样时钟超过 7 个周期，采样时钟为 16 倍波特率， DRXD 上为低表示检测到有效启动位。因此检测到一个比 7/16 长的位周期将视为有效起始位。等于或短于 7/16 将视为无效，接收器继续等待有效起始位。

检测到有效起始位后，接收器在各位理论中点处对 DRXD 采样。假设每位持续 16 个采样时钟周期 (1 位周期) 所以采样点是启动该位后第 8 个周期 (0.5 位周期)。第一个采样点是检测到启动位下降沿后的第 24 个周期 (1.5 位周期)。

每位在前一位 16 周期 (1 位周期) 后采样。

Figure 138. 起始位检测

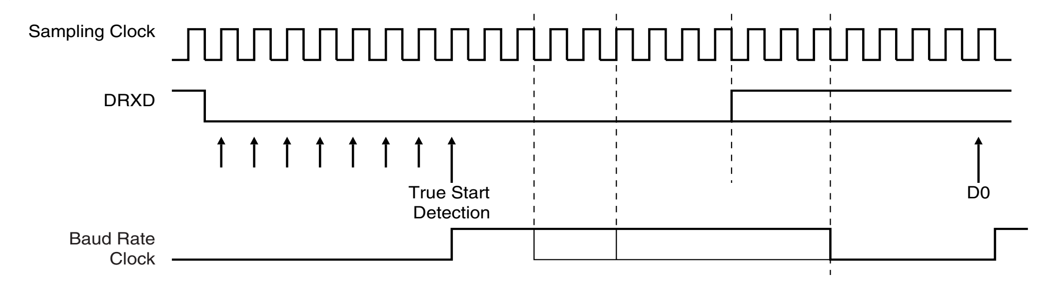
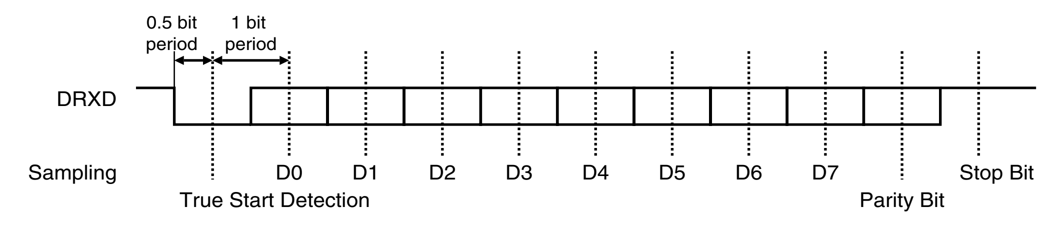


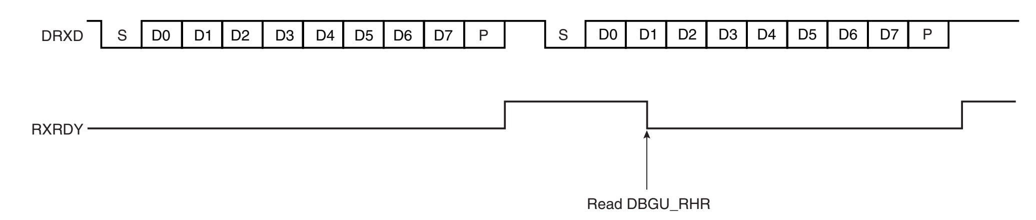
Figure 139. 字符接收



接收器就绪

接收一个完整字符后，传输到 DBGU_RHR 并将 DBGU_SR(状态寄存器) 中的 RXRDY 状态位置位。当读取接收保持寄存器 DBGU_RHR 时， RXRDY 位自动清除。

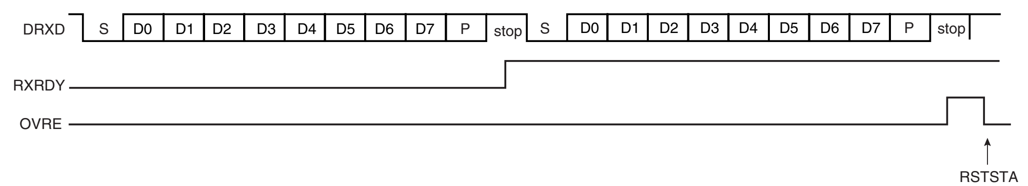
Figure 140. 接收器就绪



接收器溢出

上次传输后，若 DBGU_RHR 仍未被软件或外设数据控制器读取， RXRDY 位仍将置位并开始接收新字符，DBGU_SR 中的 OVRE 状态位置位。当软件将 DBGU_CR 的位 RSTSTA(复位状态) 写为 1 时， OVRE 位清除。

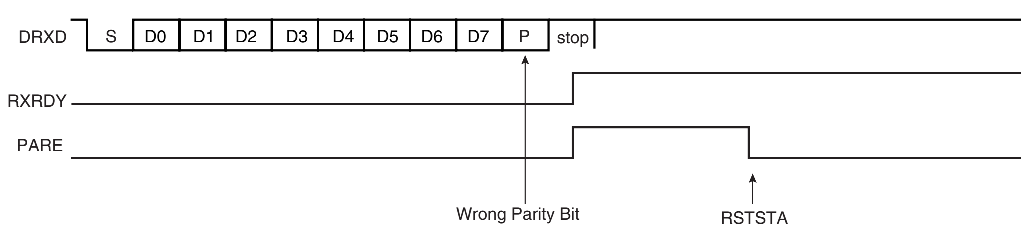
Figure 141. 接收器溢出



奇偶校验位错误

每次接收到字符后，接收器在 DBGU_MR 中的 PAR 域计算接收数据比特的奇偶校验。然后将计算结果与接收到的奇偶校验位比。若不同，在设置 RXRDY 的同时设置 DBGU_SR 中的奇偶错误校验位 PARE。当控制寄存器 DBGU_CR 的 RSTSTA 位写入 1，奇偶校验位清除。若在复位状态命令写入前收到新的字符， PARE 位仍为 1。

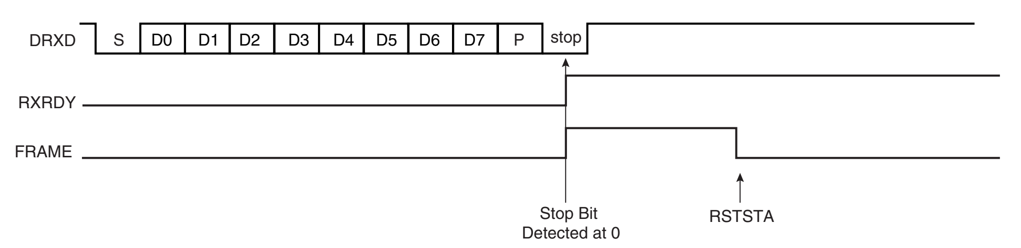
Figure 142. 奇偶校验错误



接收器帧错误

当检测到启动位，所有数据采样后产生一个字符接收信号。停止位也将采样，当检测到它为 0 时，DBGU_SR 中的 FRAME (帧错误) 位置位，同时将 RXRDY 位置位。FRAME 将保持为高直到寄存器 DBGU_CR 的 RSTSTA 位写入 1。

Figure 143. 接收器帧错误



发送器

发送器复位，使能与禁用

器件复位后，调试单元发送器禁用，在使用前必须使能。发送器可通过在控制寄存器 DBGU_CR 的 TXEN 位写入 1 使能。该命令后，发送器在开始工作前等待在发送保持寄存器 DBGU_THR 中写入符号。

程序员可在 DBGU_CR 寄存器的 TXDIS 写 1 来禁用发送器。若发送器未工作，则立即停止。但若已有字符写入移位寄存器或已写入发送保持寄存器，则在字符发送完成后再停止。

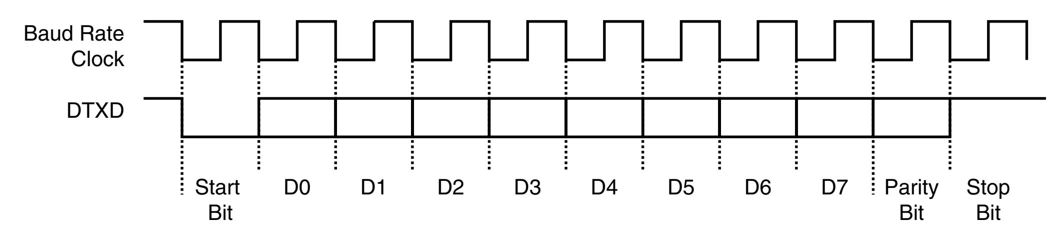
程序员也可写DBGU_CR寄存器RSTRX位为1将发送器置于复位状态。此时，发送器立即停止。

发送格式

调试单元以波特率时钟速度驱动 DTXD 引脚。驱动总线由模式寄存器定义的格式决定，而数据保存在移位寄存器中。一个电平为 0 的起始位，然后是 8 个低位在先的数据位，一个可选的奇偶校验位及一个电平为 1 的停止位连续移出。模式寄存器 DBGU_MR 的 PARE 域定义是否将奇偶校验位移出。当奇偶校验位使能，可选用奇校验、偶校验或固定空或标记位。



Figure 144. 字符传输

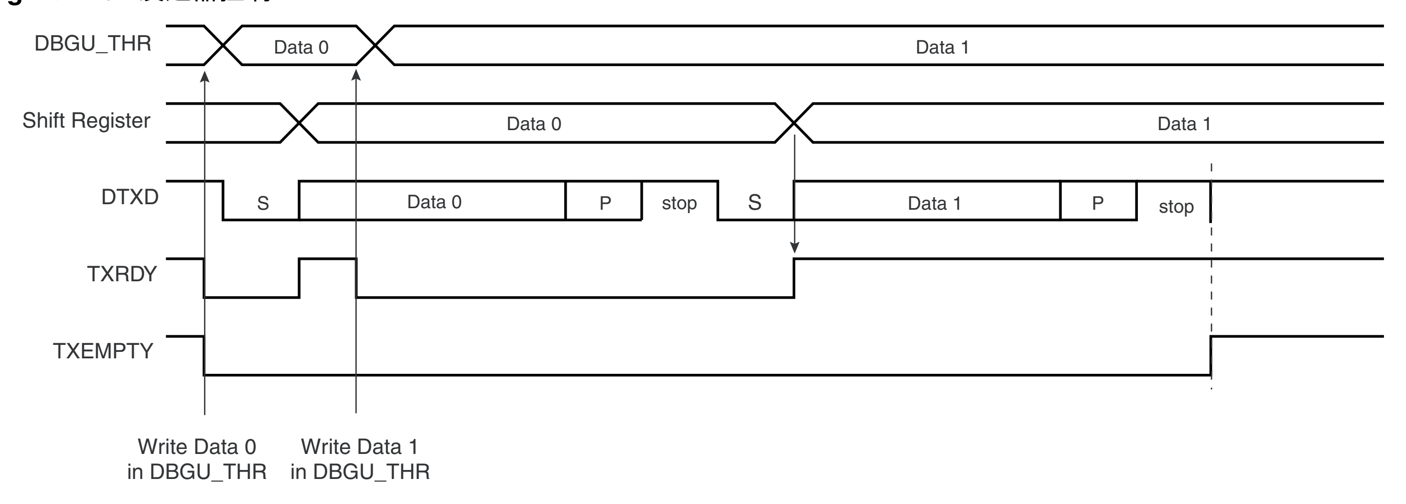


发送器控制

当发送器使能， DBGU_SR 中的 TXRDY(发送器就绪) 位置位。当程序员写发送保持寄存器 DBGU_THR 时，发送启动。在写字符发送完毕后数据由 DBGU_THR 发往移位寄存器。TXRDY 位保持为高直到第二个字符写入 DBGU_THR。首字符完成后，写入 DBGU_THR 的尾字符送入移位寄存器，且 TXRDY 重新拉高，此时保持寄存器为空。

当移位寄存器与DBGU_THR均为空时，即写入DBGU_THR中的所有字符均以处理，TXEMPTY 位在最后一个停止位结束后拉高。

Figure 145. 发送器控制



外设数据控制器

调试单元 UART 的发送器和接收器通常与外设数据控制器 (PDC) 通道连接。

外设数据控制器通道通过映射到调试单元接口偏移地址为 0x100 的寄存器编程。状态变化显示在调试状态寄存器 DBGU_SR 中并能产生一个中断。

RXRDY位触发PDC通道接收器数据传输。因此在DBGU_RHR中读取数据。TXRDY位触发PDC 通道发送器数据传输。因此在 DBGU_THR 中写入数据。

测试模式

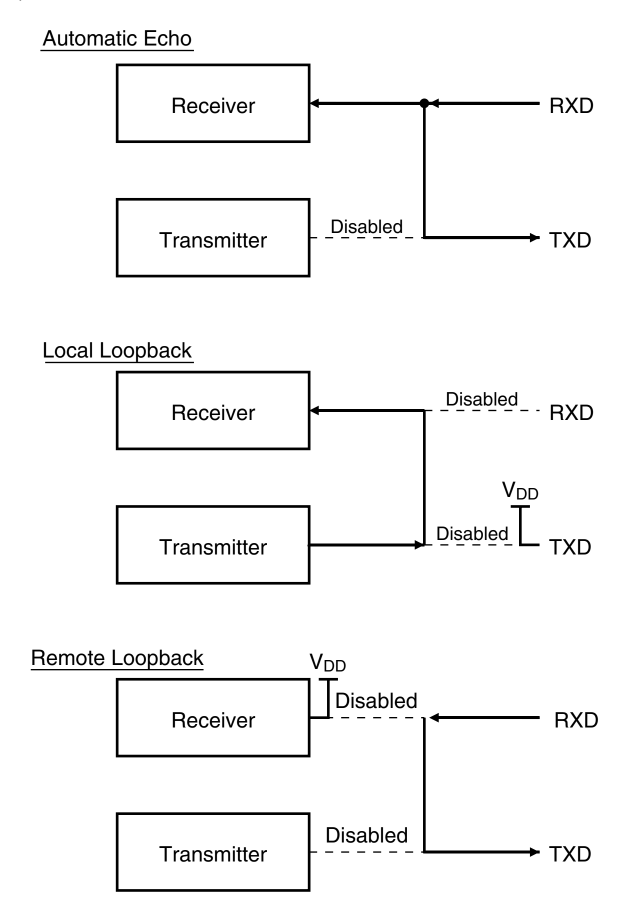
调试单元支持三种测试模式。由模式寄存器 DBGU_MR 的 CHMODE (通道模式) 域对使用哪种模式进行编程。

自动回应模式允许位转发。当 DRXD 线收到一个比特后，发送到 DTXD 线。发送器工作正常，但对 DTXD 线没有影响。

本地回环模式允许接收发送的字符。 DTXD 与 DRXD 引脚未用，发送器输出与接收器输入连接。与空闲状态相同， DRXD 引脚电平无影响， DTXD 线保持高电平。

远程回环模式直接将 DRXD 引脚与 DTXD 线连接。发送器与接收器禁用。该模式允许位重发。

Figure 146. 测试模式





调试通信通道支持

调试单元处理来自 ARM 处理器调试通信通道的 COMMRX 与 COMMTX 信号，用板上仿真器驱动。

调试通信通道包含两个由 JTAG 边 ICE 断路器及 ARM 处理器中协处理器 0 访问的寄存器。

下面的指令用来对调试通信通道进行读写：

```
MRC    p14, 0, Rd, c1, c0, 0
```

将调试通信数据寄存器中的内容返回 Rd

```
MCR    p14, 0, Rd, c1, c0, 0
```

将 Rd 中值写入调试通信数据写寄存器。

COMMRX 与 COMMTX 位，分别表示读寄存器已由调试器写入但仍未被处理器读出，及写寄存器已由处理器写入但仍未被调试器读出，位于状态寄存器 DBGU_SR 的最高两位。这两位可产生一个中断。该特性允许中断下处理运行于目标系统中的调试监控器和调试器间的调试链接。

芯片标识符

调试单元有两个芯片标识寄存器：DBGU_CIDR (芯片 ID 寄存器) 与 DBGU_EXID (扩展 ID 寄存器)。两个寄存器中均为只读的硬件固化值。首个寄存器包括下列域：

- EXT - 表示用到了扩展标识寄存器
- NVPTYP 与 NVPSIZ - 标识内置非易失性存储器类型及其大小
- ARCH - 标识内置外设
- SRAMSIZ - 标识内置 SRAM 大小
- EPROC - 标识内置 ARM 处理器
- VERSION - 给出芯片型号

第二个寄存器是由器件决定的，若 EXT 为 0，则其值为 0。

ICE 访问限制

调试单元可阻止通过 ARM 处理器的 ICE 接口对系统进行访问。通过 NTRST(DBGU_FNR) 寄存器中的 ICE 接口插入信号 NTRST 实现该特性。对 FNTRST (强制 NTRST) 位写 1 将阻止 TAP 控制器的操作。

标准器件中， FNTRST 位复位为 0，因此不会阻止 ICE 访问。

该特性主要用于不愿其芯片代码可见的 ROM 中。

调试单元用户接口

Table 65. 调试单元存储器映射

偏移	寄存器	寄存器名称	访问类型	复位值
0x0000	控制寄存器	DBGU_CR	只写	–
0x0004	模式寄存器	DBGU_MR	读 / 写	0x0
0x0008	中断使能寄存器	DBGU_IER	只写	–
0x000C	中断禁用寄存器	DBGU_IDR	只写	–
0x0010	中断屏蔽寄存器	DBGU_IMR	只读	0x0
0x0014	状态寄存器	DBGU_SR	只读	–
0x0018	接收保持寄存器	DBGU_RHR	只读	0x0
0x001C	发送保持寄存器	DBGU_THR	只写	–
0x0020	波特率发生器寄存器	DBGU_BRGR	读 / 写	0x0
0x0024 - 0x003C	保留	–	–	–
0X0040	芯片 ID 寄存器	DBGU_CIDR	只读	–
0X0044	芯片 ID 扩展寄存器	DBGU_EXID	只读	–
0X0048	强制 NTRST 寄存器	DBGU_FNR	读 / 写	0x0
0x004C - 0x00FC	保留	–	–	–
0x0100 - 0x0124	PDC 域	–	–	–



调试单元控制寄存器

寄存器名称： DBGU_CR

访问类型： 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	RSTSTA

7	6	5	4	3	2	1	0
TXDIS	TXEN	RXDIS	RXEN	RSTTX	RSTRX	–	–

- **RSTRX: 接收器复位**

0 = 无效。

1 = 接收器逻辑复位并禁用。若此时正接收字符，则停止接收。

- **RSTTX: 发送器复位**

0 = 无效。

1 = 发送器逻辑复位并禁用。若此时正发送字符，则停止发送。

- **RXEN: 接收器使能**

0 = 无效。

1 = 若 RXDIS 为 0，接收器使能。

- **RXDIS: 接收器禁用**

0 = 无效。

1 = 接收器禁用。若正在处理字符且 RSTRX 未置位，则在停止接收器前将字符处理完成。

- **TXEN: 发送器使能**

0 = 无效。

1 = 若 TXDIS 为 0，发送器使能。

- **TXDIS: 发送器禁用**

0 = 无效。

1 = 发送器禁用。若正在处理字符且已有字符写入 DBGU_THR 而 RSTTX 未置位，则在停止发送器前将两字符处理完成。

- **RSTSTA: 状态位复位**

0 = 无效。

1 = 将 DBGU_SR 寄存器中的状态位 PARE、 FRAME 及 OVRE 复位。

调试单元模式寄存器

寄存器名称： DBGU_MR

访问类型： 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
CHMODE		–	–	PAR			–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

- **PAR: 校验类型**

PAR			校验类型
0	0	0	偶校验
0	0	1	奇校验
0	1	0	空号：校验强制为 0
0	1	1	标记：校验强制为 1
1	x	x	无奇偶校验

- **CHMODE: 通道模式**

CHMODE		模式说明
0	0	正常模式
0	1	自动回应
1	0	本地回环
1	1	远程回环





调试单元中断使能寄存器

寄存器名称 : DBGU_IER

访问类型 : 只写

31	30	29	28	27	26	25	24
COMMRX	COMMTX	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	RXBUFF	TXBUFE	–	TXEMPTY	–

7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	ENDTX	ENDRX	–	TXRDY	RXRDY

- **RXRDY: 使能 RXRDY 中断**
- **TXRDY: 使能 TXRDY 中断**
- **ENDRX: 使能接收传输结束中断**
- **ENDTX: 使能发送结束中断**
- **OVRE: 使能溢出错误中断**
- **FRAME: 使能帧错误中断**
- **PARE: 使能奇偶校验错误中断**
- **TXEMPTY: 使能 TXEMPTY 中断**
- **TXBUFE: 使能缓冲器空中断**
- **RXBUFF: 使能缓冲器满中断**
- **COMMTX: 使能 COMMTX (来自 ARM) 中断**
- **COMMRX: 使能 COMMRX (来自 ARM) 中断**

0 = 无效。

1 = 使能相应中断。

调试单元中断禁用寄存器

寄存器名称 : DBGU_IDR

访问类型 : 只写

31	30	29	28	27	26	25	24
COMMRX	COMMTX	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	RXBUFF	TXBUFE	–	TXEMPTY	–

7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	ENDTX	ENDRX	–	TXRDY	RXRDY

- **RXRDY: 禁用 RXRDY 中断**
- **TXRDY: 禁用 TXRDY 中断**
- **ENDRX: 禁用接收传输结束中断**
- **ENDTX: 禁用发送结束中断**
- **OVRE: 禁用溢出错误中断**
- **FRAME: 禁用帧错误中断**
- **PARE: 禁用奇偶校验错误中断**
- **TXEMPTY: 禁用 TXEMPTY 中断**
- **TXBUFE: 禁用缓冲器空中断**
- **RXBUFF: 禁用缓冲器满中断**
- **COMMTX: 禁用 COMMTX (来自 ARM) 中断**
- **COMMRX: 禁用 COMMRX (来自 ARM) 中断**

0 = 无效。

1 = 禁用相应中断。



调试单元中断屏蔽寄存器

寄存器名称 : DBGU_IMR

访问类型 : 只读

31	30	29	28	27	26	25	24
COMMRX	COMMTX	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	RXBUFF	TXBUFE	–	TXEMPTY	–

7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	ENDTX	ENDRX	–	TXRDY	RXRDY

- **RXRDY: 屏蔽 RXRDY 中断**
- **TXRDY: 屏蔽 TXRDY 中断**
- **ENDRX: 屏蔽接收传输结束中断**
- **ENDTX: 屏蔽发送结束中断**
- **OVRE: 屏蔽溢出错误中断**
- **FRAME: 屏蔽帧错误中断**
- **PARE: 屏蔽奇偶校验错误中断**
- **TXEMPTY: 屏蔽 TXEMPTY 中断**
- **TXBUFE: 屏蔽缓冲器空中断**
- **RXBUFF: 屏蔽缓冲器满中断**
- **COMMTX: 屏蔽 COMMTX 中断**
- **COMMRX: 屏蔽 COMMRX 中断**

0 = 相应中断禁用。

1 = 相应中断使能。

调试单元状态寄存器

寄存器名称： DBGU_SR

访问类型： 只读

31	30	29	28	27	26	25	24
COMMRX	COMMTX	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	RXBUFF	TXBUFE	–	TXEMPTY	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PARE	FRAME	OVRE	ENDTX	ENDRX	–	TXRDY	RXRDY

- **RXRDY: 接收器就绪**

0 = 上次读 DBGU_RHR 后未收到字符或接收器禁用。

1 = 至少收到一个完整的字符传输到 DBGU_RHR 并还未读。

- **TXRDY: 发送器就绪**

0 = 已将字符写入 DBGU_THR 但仍未发送到移位寄存器或发送器禁用。

1 = DBGU_THR 中没有未发送到移位寄存器中的字符。

- **ENDRX: 接收器传输结束**

0 = 接收器外设数据控制器通道中的传输结束信号无效。

1 = 接收器外设数据控制器通道中的传输结束信号有效。

- **ENDTX: 发送器传输结束**

0 = 发送器外设数据控制器通道中的传输结束信号无效。

1 = 发送器外设数据控制器通道中的传输结束信号有效。

- **OVRE: 溢出错误**

0 = 上次 RSTSTA 后无溢出错误。

1 = 上次 RSTSTA 后至少发生一次溢出错误。

- **FRAME: 帧错误**

0 = 上次 RSTSTA 后无帧错误。

1 = 上次 RSTSTA 后至少发生一次帧错误。

- **PARE: 奇偶校验错误**

0 = 上次 RSTSTA 后无奇偶校验错误。

1 = 上次 RSTSTA 后至少发生一次奇偶校验错误。

- **TXEMPTY: 发送器空**

0 = DBGU_THR 中有字符，或发送器正在处理字符或发送器禁用。

1 = DBGU_THR 中无字符且发送器没有处理字符。

- **TXBUFE: 发送缓冲器空**

0 = 来自发送器 PDC 通道的缓冲器空信号无效。

1 = 来自发送器 PDC 通道的缓冲器空信号有效。

- **RXBUFF: 接收缓冲器满**

0 = 来自接收器 PDC 通道的缓冲器满信号无效。

1 = 来自接收器 PDC 通道的缓冲器满信号有效。



- **COMMTX: 调试通信通道写状态**

0 = 来自 ARM 处理器的 COMMTX 无效。

1 = 来自 ARM 处理器的 COMMTX 有效。

- **COMMRX: 调试通信通道读状态**

0 = 来自 ARM 处理器的 COMMRX 无效。

1 = 来自 ARM 处理器的 COMMRX 有效。

调试单元接收器保持寄存器

寄存器名称 : DBGU_RHR

访问类型 : 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RXCHR							

- **RXCHR:** 收到的字符

若 RXRDY 置位，则为最后收到的字符。

调试单元发送保持寄存器

寄存器名称 : DBGU_THR

访问类型 : 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TXCHR							

- **TXCHR:** 要发送的字符

若 TXRDY 未置位，为下一个要发送的字符。

调试单元波特率发生器寄存器

寄存器名称： DBGU_BRGR

访问类型： 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
CD							

7	6	5	4	3	2	1	0
CD							

- **CD: 时钟分频器**

CD	波特率时钟
0	禁用
1	MCK
2 到 65535	MCK / (CD x 16)

调试单元芯片 ID 寄存器

寄存器名称： DBGU_CIDR

访问类型： 只读

31	30	29	28	27	26	25	24
EXT	NVPTYP			ARCH			

23	22	21	20	19	18	17	16
ARCH				SRAMSIZ			

15	14	13	12	11	10	9	8
0	0	0	0	NVPSIZ			

7	6	5	4	3	2	1	0
EPROC			VERSION				

- **VERSION:** 器件版本
- **EPROC:** 内置处理器

EPROC			处理器
0	0	1	ARM946ES
0	1	0	ARM7TDMI
1	0	0	ARM920T

- **NVPSIZ:** 非易失性程序存储器大小

NVPSIZ				大小
0	0	0	0	无
0	0	0	1	8K 字节
0	0	1	0	16K 字节
0	0	1	1	32K 字节
0	1	0	0	保留
0	1	0	1	64K 字节
0	1	1	0	保留
0	1	1	1	128K 字节
1	0	0	0	保留
1	0	0	1	256K 字节
1	0	1	0	保留
1	0	1	1	保留
1	1	0	0	保留
1	1	0	1	保留
1	1	1	0	保留
1	1	1	1	保留



- **SRAMSIZ: 内部 SRAM 大小**

SRAMSIZ				大小
0	0	0	0	保留
0	0	0	1	1K 字节
0	0	1	0	2K 字节
0	0	1	1	保留
0	1	0	0	保留
0	1	0	1	4K 字节
0	1	1	0	保留
0	1	1	1	保留
1	0	0	0	8K 字节
1	0	0	1	16K 字节
1	0	1	0	32K 字节
1	0	1	1	64K 字节
1	1	0	0	128K 字节
1	1	0	1	256K 字节
1	1	1	0	96K 字节
1	1	1	1	512K 字节

- **ARCH: 结构标识符**

ARCH		
Hex	Dec	结构
0x40	0100 0000	AT91x40 系列
0x63	0110 0011	AT91x63 系列
0x55	0101 0101	AT91x55 系列
0x42	0100 0010	AT91x42 系列
0x92	1001 0010	AT91x92 系列
0x34	0011 0100	AT91x34 系列

- **NVPTYP: 非易失性程序存储器类型**

NVPTYP			存储器
0	0	0	ROM
0	0	1	ROMless 或片上 Flash
1	0	0	SRAM 仿真 ROM

- **EXT: 扩展标志**

0 = 芯片 ID 在无扩展的单寄存器中定义。

1 = 存在扩展芯片 ID。

调试单元芯片 ID 扩展寄存器

寄存器名称 : DBGU_EXID

访问类型 : 只读

31	30	29	28	27	26	25	24
EXID							

23	22	21	20	19	18	17	16
EXID							

15	14	13	12	11	10	9	8
EXID							

7	6	5	4	3	2	1	0
EXID							

- **EXID: 芯片 ID 扩展**

若 DBGU_CIDR 中 EXT 位为 0，则其值为 0。

调试单元强制 NTRST 寄存器

寄存器名称 : DBGU_FNR

访问类型 : 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	FNTRST

- **FNTRST: 强制 NTRST**

0 = ARM 处理器的 TAP 控制器 NTRST 由 NTRST 引脚驱动。

1 = ARM 处理器的 TAP 控制器 NTRST 为低。

并行输入 / 输出控制器 (PIO)

概述

并行输入 / 输出控制器 (PIO) 管理高达 32 全可编程输入 / 输出线。每个 I/O 线可作为通用功能 I/O 或分配给一个内置外设。这将确保产品引脚的有效优化。

每个 I/O 线均有一个与 32 位宽的用户接口的 32 位寄存器相关的位序号。

PIO 控制器的每个 I/O 线有如下特性：

- 任意 I/O 线上的输入改变中断将使能电平变化检测。
- 毛刺滤波器可拒绝低于 1.5 时钟周期的脉冲。
- 类似于开漏 I/O 线的多驱动能力。
- 控制 I/O 线上拉。
- 输入可见，输出控制。

PIO 控制器还有同步输出特性，可在单写操作中提供高达 32 位的数据输出。

PIO 重要特性如下：

- 高达 32 个可编程 I/O 线
- 通过寄存器设置 / 清零全可编程
- 每条 I/O 线复用两个外设功能
- 每条 I/O 线 (不管是分配给外设还是用作普通功能 I/O)
  – 输入变化中断
  – 毛刺滤波器
  – 可漏时多驱动选项使能驱动
  – 每条 I/O 线上可编程上拉
  – 引脚数据状态寄存器，可在任意时刻提供引脚的可见状态
- 同步输出，提供单步写操作对几个 I/O 线的设置与清零

方框图

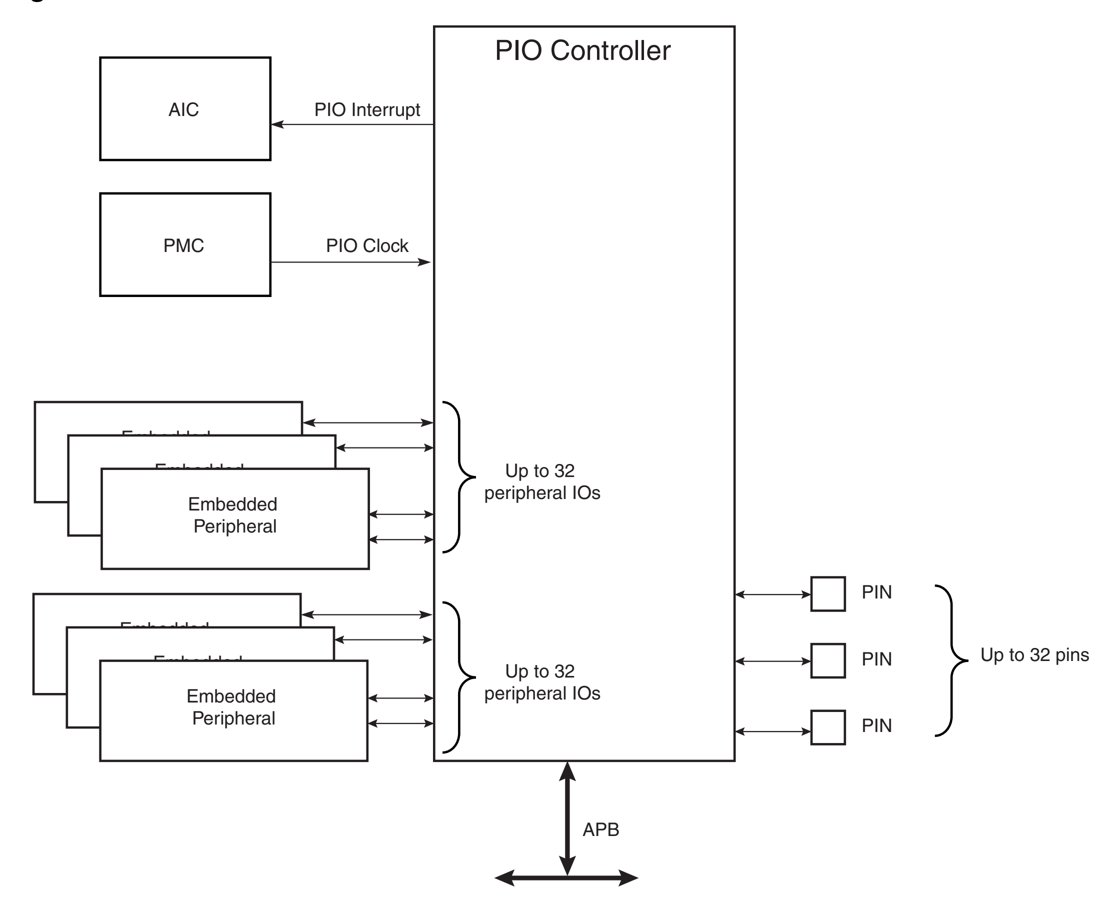
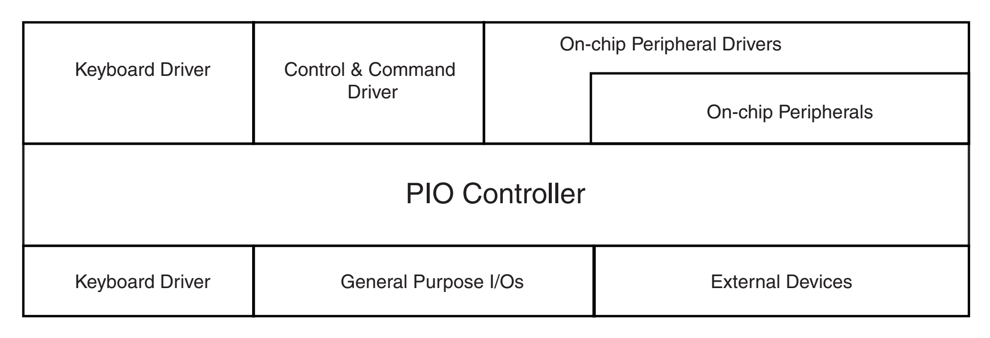


Figure 147. 方框图

Figure 148. 应用框图



附属产品

引脚复用

每个引脚可配置为通用功能 I/O 线或与一个或两个外设 I/O 复用的 I/O 线。由于复用是硬件定义的，因此硬件设计师与程序员必须根据应用需要小心决定 PIO 控制器配置。若只作为通用功能 I/O 线使用，即没有与任何外设 I/O 复用，对分配给外设的 PIO 控制器编程无效，且仅 PIO 控制器可控制如何驱动引脚。

外部中断线

中断信号 FIQ 及 IRQ0 到 IRQn 一般通过 PIO 控制器复用。但是，由于 PIO 控制器对于输入无效且中断线 (FIQ 或 IRQ) 仅作为输入，因此不必为中断分配 I/O 线。

电源管理

电源管理控制器控制 PIO 控制器时钟以节省功耗。对用户接口寄存器写入时不需要将 PIO 控制器时钟使能。即配置 I/O 线 不需要将 PIO 控制器时钟使能。

但当时钟禁用时，PIO 控制器某些功能将不可用。输入变化中断与读引脚电平就需要时钟有效。

硬件复位后，默认将 PIO 时钟禁用。

在访问输入线信息前必须配置电源管理控制器。

中断产生

对于中断处理，认为 PIO 控制器为用户外设。即 PIO 控制器中断线连接在中断源 2 到 31 间。参见 PIO 控制器外设标识符以确定分配给 PIO 控制器的中断源。

只有当 PIO 控制器时钟使能时才能产生 PIO 控制器中断。



功能说明

PIO 控制器有多达 32 的全可编程 I/O 线。Figure 149 给出了大部分与每个 I/O 相关的控制逻辑。

Figure 149. I/O Line Control Logic

上拉电阻控制

每个 I/O 线都有一个内置上拉电阻。阻值约为 100 kΩ (详见产品电气特性)。通过对 PIO_PUER (上拉使能电阻) 及 PIO_PUDR (上拉禁用电阻) 写入可使能或禁用上拉电阻。对这些寄存器写入的结果是置位或清零 PIO_PUSR(上拉状态寄存器) 中的相应位。 PIO_PUSR 读出值为 1 表示上拉禁用，读出值为 0 则表示上拉使能。

对上拉电阻的控制与 I/O 线配置无关。

复位后，所有上拉使能，即 PIO_PUSR 复位值为 0x0。

I/O 线或外设功能选择

当引脚复用一或两个外设功能，通过 PIO_PER (PIO 使能寄存器) 及 PIO_PDR (PIO 禁用寄存器) 控制功能选择。PIO_PSR (PIO 状态寄存器) 为寄存器置位或清零的结果，并用来指示引脚是由相应的外设还是由 PIO 控制器来控制。值为 0 时表示引脚由 PIO_ABSR (AB 选择状态寄存器) 相应的片上外设选择来控制；值为 1 表示由 PIO 控制器来控制。

若引脚作为通用功能 I/O 线使用 (未与片上外设复用) ,PIO_PER 与 PIO_PDR 无效且 PIO_PSR 相应位返回 1。

通常情况下，复位后， I/O 线由 PIO 控制器来控制，即 PIO_PSR 复位值为 1。但某些情况下，PIO要由外设控制(如复位后存储器片选必须置为无效或外部存储器导出时地址线必须为低)。因此， PIO_PSR 复位值由产品定义，取决于器件复用。

外设 A 或 B 选择

PIO控制器可在单引脚上提供两外设功能复用。通过对PIO_ASR (A 选择寄存器)与PIO_BSR (B 选择寄存器) 写入来选择。 PIO_ABSR 指出当前选择的外设线。对于每个引脚，相应位为 0 表示选择外设 A ；相应位为 1 表示选择外设 B。

注意外设线 A 与 B 复用仅对输出线有影响。外设输入线只与引脚输入连接。

复位后 PIO_ABSR 为 0，即表示所有 PIO 线分配给外设 A。但是在 I/O 线模式下， PIO 控制器复位后，外设 A 通常不会驱动该引脚。

不管引脚配置，对 PIO_ASR 及 PIO_BSR 写入时将影响 PIO_ABSR 值。但是在给引脚分配外设功能时除需写 PIO_PDR 外还需写入相应外设选择寄存器 (PIO_ASR 或 PIO_BSR)。

输出控制

当 I/O 线分配为外设功能后，即 PIO_PSR 相应位为 0，由外设控制 I/O 线驱动。外设 A 或 B 由 PIO_ABSR 值确定是否驱动引脚。

当 I/O 线由 PIO 控制器控制，引脚可配置为驱动。通过写 PIO_OER (输出使能寄存器) 及 PIO_PDR (输出禁用寄存器)实现。写操作结果由PIO_OSR (输出状态寄存器)中得到。当该寄存器中位值为 0，相应 I/O 线只作为输入使用；当该位值为 1，相应 I/O 线由 PIO 控制器驱动。

通过写 PIO_SODR (置位输出数据寄存器) 与 PIO_CODR (清零输出数据寄存器) , 可将 I/O 改为电平驱动。写操作对 PIO_ODSR (输出数据状态寄存器) 分别置位与清零，表示在 I/O 线上的数据驱动。不管引脚配置为 PIO 控制器控制还是分配给外设功能 , 写 PIO_OER 与 PIO_ODR 将导致 PIO_OSR 变化。这使得可通过 PIO 控制器管理优先设置 I/O 线。

类似的，写 PIO_SODR 与 PIO_CODR 将影响 PIO_ODSR。这对于定义 I/O 线第一级驱动非常重要。

同步数据输出

对 PIO_SODR 与 PIO_CODR 寄存器执行写操作需执行多条指令以定义多位值。对 8 位端口的 I/O 线的设置与清零不能同时进行，这将会限制 PIO 控制器的应用。

为避免发生这些不便， PIO 控制器提供同步数据输出特性来在单写操作中清零与置位多个 I/O 线。可通过设置 PIO_OWER (输出写使能寄存器)、 PIO_OWDR (输出写禁用寄存器) 与 PIO_OWSR (输出写状态寄存器) 来批准对 PIO_ODSR (输出数据状态寄存器) 的写入。PIO_OWSR 寄存器值可由用户对 PIO_OWER 与 PIO_OWDR 写入来定义。PIO 控制器将其作为 PIO_ODSR 写认证屏蔽。认证用户定义的写入 PIO_ODSR 的位数非常有用 , 它保证了写寄存器时未认证位不会改变，也避免了读 - 修改 - 写操作的时间消耗。





复位后， PIO_OWSR 复位值为 0x0，所有 I/O 线上禁用同步数据输出。

多驱动控制 (开漏)

通过使用多驱动控制，每个 I/O 可在开漏时独立编程。该特性允许多个驱动器连接到 I/O 线上，I/O 线仅可由每个器件驱动为低。一般需要一个外部上拉电阻 (或使能内部上拉电阻) 以确保线上高电平。

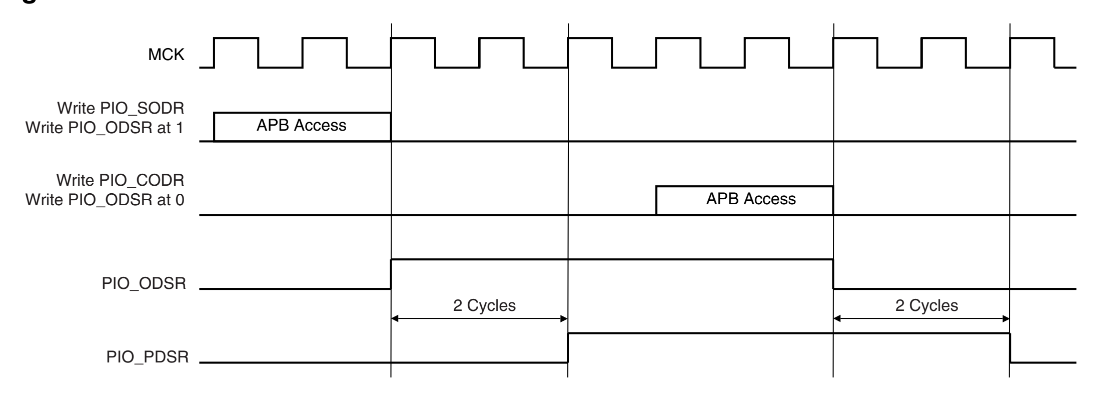
多驱动特性由 PIO_MDER (多驱动器使能寄存器) 及 PIO_MDDR (多驱动器禁用寄存器) 来控制。可选择多驱动的 I/O 线由 PIO 控制还是分配给外设功能。 PIO_MDSR (多驱动状态寄存器) 表示配置引脚支持外部驱动器。

复位后，由于 PIO_MDSR 复位值为 0x0，所有引脚多驱动特性禁用。

输出线时序

Figure 150给出写PIO_SODR或PIO_CODR或直接写PIO_ODSR的驱动输出。最后一种情况只有在 PIO_OWSR 中相应位置位时才有效。 Figure 150 给出了 PIO_PDSR 反馈有效的时刻。

Figure 150. 输出线时序



输入

每个 I/O 线电平可通过 PIO_PDSR (外设数据状态寄存器) 读出。该寄存器指示 I/O 线电平，不管它是仅作为输入还是由 PIO 控制器或外设驱动。

对 I/O 线电平读取时需要将 PIO 控制器时钟使能，否则 PIO_PDSR 读到的是时钟禁用时的 I/O 线电平。

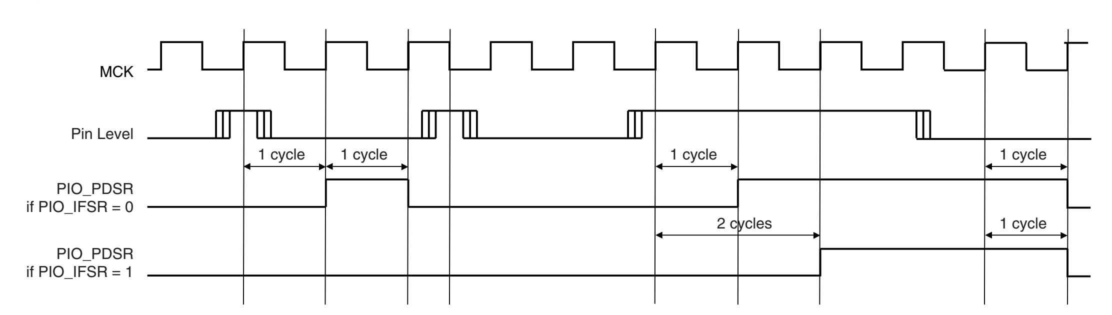
输入毛刺滤波

每个 I/O 线可选的输入毛刺滤波器可独立编程。当毛刺滤波器使能，自动滤除小于 1/2 主机时钟 (MCK) 周期的毛刺，而当大于 1 个主机时钟周期的信号将通过。对于在 1/2 到 1 个主机时钟周期间的信号则由其出现的精确时间决定是否能通过。因此对于信号必须超过 1 个主机时钟周期，而毛刺则要小于 1/2 主机时钟周期。若在上升沿前引脚电平变化，则滤波器引入 1 个主机时钟周期延迟。但若在下降沿前出现引脚电平变化，则不会有延迟，详见 Figure 151。

毛刺滤波器由 PIO_IFER (输入滤波器使能寄存器)、 PIO_IFDR (输入滤波器禁用寄存器) 及 PIO_IFSR (输入滤波器状态寄存器) 控制。分别对 PIO_IFER 及 PIO_IFDR 写入将置位或清零 PIO_IFSR 中位。最后一个寄存器使能 I/O 线上的毛刺滤波器。

当毛刺滤波器使能，不会修改外设输入。它仅在读 PIO_PDSR 值及输入改变中断检测中起作用。毛刺滤波器要求 PIO 控制器时钟使能。

Figure 151. 输入毛刺滤波器时序



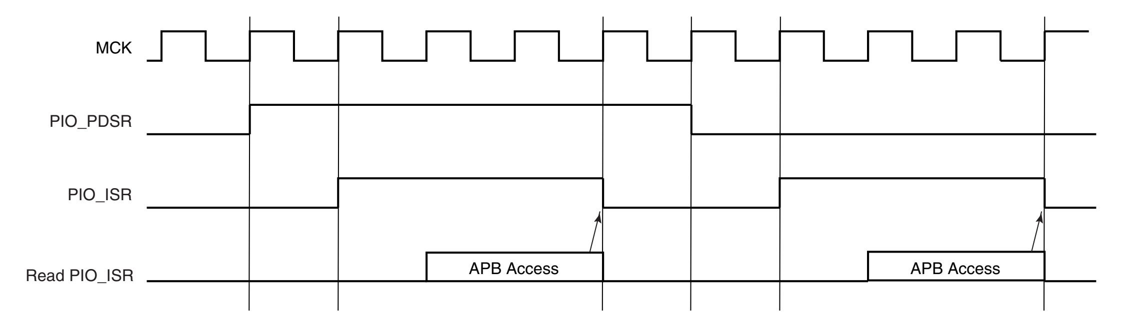
输入变化中断

当 PIO 控制器检测到 I/O 线上输入变化时，它可编程产生中断。输入变化中断由 PIO_IER（中断使能寄存器）与 PIO_IDR（中断禁用寄存器）的写入来控制，两寄存器分别通过置位与清零 PIO_IMR（中断屏蔽寄存器）中的相应位来使能或禁用输入变化中断。由于输入变化检测只能通过比较连续两个输入采样值得到，因此 PIO 控制器时钟必须使能。不管 I/O 线如何配置，输入变化中断有效 。

当检测到 I/O 线上的输入变化，PIO_ISR（中断状态寄存器）中的相应位置位。若 PIO_IMR 相应位置位，插入 PIO 控制器中断。32 通道中断信号线或后产生一个单中断信号，送往高级中断控制器。

当软件读取 PIO_ISR 时，所有中断自动清零。即当读取 PIO_ISR 时，所有挂起的中断必须处理。

Figure 152. 输入变化中断时序



I/O 线编程示例

Table 66 给出的编程示例定义以下配置。

- I/O 线 0 到 3 上 4 位输出端口 (应在单写操作中写入) ，开漏 ，有上拉电阻
- I/O 线 4 到 7 上 4 个输出信号 (例如驱动 LED) ，驱动高低电平 ，无上拉电阻
- I/O线8到11上4个输入信号(例如读取按钮操作状态) ,有上拉电阻、毛刺滤波器即输入变化中断
- I/O 线 12 到 15 上 4 个输入信号读取外部器件状态 (轮询 ,因此无输入变化中断) ,无上拉电阻 ，无毛刺滤波器
- I/O 线 16 到 19 分配给外设 A ，有上拉电阻
- I/O 线 20 到 23 分配给外设 B ，无上拉电阻
- I/O 线 24 到 27 分配给外设 A ，有输入变化中断 ，有上拉电阻

Table 66. 编程示例

寄存器	写入值
PIO_PER	0x0000 FFFF
PIO_PDR	0x0FFF 0000
PIO_OER	0x0000 00FF
PIO_ODR	0x0FFF FF00
PIO_IFER	0x0000 0F00
PIO_IFDR	0x0FFF F0FF
PIO_SODR	0x0000 0000
PIO_CODR	0x0FFF FFFF
PIO_IER	0x0F00 0F00
PIO_IDR	0x00FF F0FF
PIO_MDER	0x0000 000F
PIO_MDDR	0x0FFF FFF0
PIO_PUDR	0x00F0 00F0
PIO_PUER	0x0F0F FF0F
PIO_ASR	0x0F0F 0000
PIO_BSR	0x00F0 0000
PIO_OWER	0x0000 000F
PIO_OWDR	0x0FFF FFF0

并行输入 / 输出控制器 (PIO) 用户接口

PIO 控制器控制的每个 I/O 线都与 PIO 控制器用户接口寄存器中的某一位相关。每个寄存器有 32 位宽。若一个并行 I/O 线未定义，对相应位写入无效。未定义位读为零。若 I/O 线未与外设复用， I/O 线由 PIO 控制器控制且 PIO_PSR 返回 1。

Table 67. PIO 寄存器映射

偏移	寄存器	名称	访问类型	复位值
0x0000	PIO 使能寄存器	PIO_PER	只写	–
0x0004	PIO 禁用寄存器	PIO_PDR	只写	–
0x0008	PIO 状态寄存器 (1)	PIO_PSR	只读	0x0000 0000
0x000C	保留			
0x0010	输出使能寄存器	PIO_OER	只写	–
0x0014	输出禁用寄存器	PIO_ODR	只写	–
0x0018	输出状态寄存器	PIO_OSR	只读	0x0000 0000
0x001C	保留			
0x0020	毛刺输入滤波器使能寄存器	PIO_IFER	只写	–
0x0024	毛刺输入滤波器禁用寄存器	PIO_IFDR	只写	–
0x0028	毛刺输入滤波器状态寄存器	PIO_IFSR	只读	0x0000 0000
0x002C	保留			
0x0030	PIO 置位输出数据寄存器	PIO_SODR	只写	–
0x0034	PIO 清零输出数据寄存器	PIO_CODR	只写	–
0x0038	PIO 输出数据状态寄存器 (2)	PIO_ODSR	只读	0x0000 0000
0x003C	PIO 引脚数据状态寄存器 (3)	PIO_PDSR	只读	
0x0040	PIO 中断使能寄存器	PIO_IER	只写	–
0x0044	PIO 中断禁用寄存器	PIO_IDR	只写	–
0x0048	PIO 中断屏蔽寄存器	PIO_IMR	只读	0x0000 0000
0x004C	PIO 中断状态寄存器 (4)	PIO_ISR	只读	0x0000 0000
0x0050	PIO 多驱动使能寄存器	PIO_MDER	只写	–
0x0054	PIO 多驱动禁用寄存器	PIO_MDDR	只写	–
0x0058	PIO 多驱动状态寄存器	PIO_MDSR	只读	0x0000 0000
0x005C	保留			
0x0060	PIO 上拉禁用寄存器	PIO_PUDR	只写	–
0x0064	PIO 上拉使能寄存器	PIO_PUER	只写	–
0x0068	PIO 上拉状态寄存器	PIO_PUSR	只读	0x0000 0000
0x006C	保留			

Table 67. PIO 寄存器映射

偏移	寄存器	名称	访问类型	复位值
0x0070	PIO 外设 A 选择寄存器 (5)	PIO_ASR	只写	–
0x0074	PIO 外设 B 选择寄存器 (5)	PIO_BSR	只写	–
0x0078	PIO AB 状态寄存器 (5)	PIO_ABSR	只读	0x0000 0000
0x007C-0x009C	保留			
0x00A0	PIO 输出写使能	PIO_OWER	只写	–
0x00A4	PIO 输出写禁用	PIO_OWDR	只写	–
0x00A8	PIO 输出写状态寄存器	PIO_OWSR	只读	0x0000 0000
0x00AC	保留			

Notes:
1. PIO_PSR 复位值取决于产品。
2. PIO_ODSR 根据 PIO_OWSR I/O 线不同为只读或读 / 写。
3. PIO_PDSR 复位值取决于 I/O 线电平。
4. PIO_ISR 复位值为 0x0。但当首次读该寄存器时可能会由于发生输入变化而导致读出一个不同的值。
5. 仅该系列寄存器可通过向第一个寄存器写 1 清除状态，对第二个寄存器写 1 置位状态。

PIO 使能寄存器

寄存器名称： PIO_PER

访问类型： 只写

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: PIO 使能**

0 = 无效。

1 = 使能 PIO 来控制相应引脚 (禁用引脚外设控制)。

PIO 禁用寄存器

寄存器名称： PIO_PDR

访问类型： 只写

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: PIO 禁用**

0 = 无效。

1 = 禁用 PIO 控制相应引脚 (使能引脚外设控制)。



PIO 状态寄存器

寄存器名称 : PIO_PSR

访问类型 : 只读

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: PIO 状态**

0 = 相应的 I/O 线上 PIO 无效 (外设激活)。

1 = 相应的 I/O 线上 PIO 有效 (外设无效)。

PIO 输出使能寄存器

寄存器名称 : PIO_OER

访问类型 : 只写

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: 输出使能**

0 = 无效。

1 = 使能 I/O 线上输出。

PIO 输出禁用寄存器

寄存器名称： PIO_ODR

访问类型： 只写

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: 输出禁用**

0 = 无效。

1 = 禁用 I/O 线上输出。

PIO 输出状态寄存器

寄存器名称： PIO_OSR

访问类型： 只读

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: 输出状态**

0 = I/O 线为纯输入。

1 = I/O 线输出使能。

PIO 输入滤波器使能寄存器

寄存器名称： PIO_IFER

访问类型： 只写

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: 输入滤波器使能**

0 = 无效。

1 = 使能 I/O 线上输入毛刺滤波器。

PIO 输入滤波器禁用寄存器

寄存器名称： PIO_IFDR

访问类型： 只写

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: 输入滤波器禁用**

0 = 无效。

1 = 禁用 I/O 线上输入毛刺滤波器。

PIO 输入滤波器状态寄存器

寄存器名称 : PIO_IFSR

访问类型 : 只读

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
P23	P22	P21	P20	P19	P18	P17	P16
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
P15	P14	P13	P12	P11	P10	P9	P8
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: 输入滤波器状态**

0 = I/O 线上输入毛刺滤波器禁用。

1 = I/O 线上输入毛刺滤波器使能。

PIO 置位输出数据寄存器

寄存器名称 : PIO_SODR

访问类型 : 只写

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
P23	P22	P21	P20	P19	P18	P17	P16
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
P15	P14	P13	P12	P11	P10	P9	P8
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: 置位输出数据**

0 = 无效。

1 = 设置在 I/O 线上驱动的数据。

PIO 输出数据清零寄存器

寄存器名称 : PIO_CODR

访问类型 : 只写

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: 设置输出数据**

0 = 无效。

1 = 清除在 I/O 线上驱动的数据。

PIO 输出数据状态寄存器

寄存器名称 : PIO_ODSR

访问类型 : 只读或读 / 写

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: 输出数据状态**

0 = 驱动到 I/O 线上的数据为 0。

1 = 驱动到 I/O 线上的数据为 1。

PIO 引脚数据状态寄存器

寄存器名称 : PIO_PDSR

访问类型 : 只读

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: 输出数据状态**

0 = I/O 线电平为 0。

1 = I/O 线电平为 1。

PIO 中断使能寄存器

寄存器名称 : PIO_IER

访问类型 : 只写

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: 输入变化中断使能**

0 = 无效。

1 = 使能 I/O 线上输入变化中断。





PIO 中断禁用寄存器

寄存器名称 : PIO_IDR

访问类型 : 只写

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: 输入变化中断禁用**

0 = 无效。

1 = 禁用 I/O 线上输入变化中断。

PIO 中断屏蔽寄存器

寄存器名称 : PIO_IMR

访问类型 : 只读

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: 输入变化中断屏蔽**

0 = I/O 线上输入变化中断禁用。

1 = I/O 线上输入变化中断使能。

PIO 中断状态寄存器

寄存器名称 : PIO_IMR

访问类型 : 只读

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
P23	P22	P21	P20	P19	P18	P17	P16
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
P15	P14	P13	P12	P11	P10	P9	P8
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: 输入变化中断状态**

0 = 上次 PIO_ISR 读后或复位后 , I/O 线上未检测到输入变化。

1 = 上次 PIO_ISR 读后或复位后 , I/O 线上至少检测到一次输入变化。

PIO 多驱动使能寄存器

寄存器名称 : PIO_MDER

访问类型 : 只写

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
P23	P22	P21	P20	P19	P18	P17	P16
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
P15	P14	P13	P12	P11	P10	P9	P8
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: 多驱动使能**

0 = 无效。

1 = 使能 I/O 线上多驱动。



PIO 多驱动禁用寄存器

寄存器名称 : PIO_MDDR

访问类型 : 只写

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: 多驱动禁用**

0 = 无效。

1 = 禁用 I/O 线上多驱动。

PIO 多驱动状态寄存器

寄存器名称 : PIO_MDSR

访问类型 : 只读

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: 多驱动状态**

0 = I/O 线上多驱动禁用。引脚驱动为高低电平。

1 = I/O 线上多驱动使能。引脚仅驱动为低电平。

PIO 上拉禁用寄存器

寄存器名称： PIO_PUDR

访问类型： 只写

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: 上拉禁用**

0 = 无效。

1 = 禁用 I/O 线上拉电阻。

PIO 上拉使能寄存器

寄存器名称： PIO_PUER

访问类型： 只写

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: 上拉使能**

0 = 无效。

1 = 使能 I/O 线上拉电阻。



PIO 上拉状态寄存器

寄存器名称 : PIO_PUSR

访问类型 : 只读

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31:** 上拉状态

0 = I/O 线上拉电阻使能。

1 = I/O 线上拉电阻禁用。

PIO 外设 A 选择寄存器

寄存器名称 : PIO_ASR

访问类型 : 只写

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31:** 外设 **A** 选择

0 = 无效。

1 = I/O 线分配给外设 A。

PIO 外设 B 选择寄存器

寄存器名称 : PIO_BSR

访问类型 : 只写

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: 外设 B 选择**

0 = 无效。

1 = I/O 线分配给外设 B。

PIO 外设 A B 状态寄存器

寄存器名称 : PIO_ABSR

访问类型 : 只读

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: 外设 A B 状态**

0 = I/O 线分配给外设 A。

1 = I/O 线分配给外设 B。



PIO 输出写使能寄存器

寄存器名称 : PIO_OWER

访问类型 : 只写

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: 输出写使能**

0 = 无效。

1 = 使能 I/O 线对 PIO_ODSR 写。

PIO 输出写禁用寄存器

寄存器名称 : PIO_OWDR

访问类型 : 只写

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: 输出写禁用**

0 = 无效。

1 = 禁用 I/O 线对 PIO_ODSR 写。

PIO 输出写状态寄存器

寄存器名称 : PIO_OWSR

访问类型 : 只读

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
P23	P22	P21	P20	P19	P18	P17	P16
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
P15	P14	P13	P12	P11	P10	P9	P8
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: 输出写状态**

0 = 写 PIO_ODSR 不影响 I/O 线。

1 = 写 PIO_ODSR 影响 I/O 线。

串行外设接口 (SPI)

概述

串行外设接口 (SPI) 电路是同步串行数据链接，在主机或从机模式下提供于外部器件的特性。若外部处理器与系统连接，它还使能处理器间通信。

串行外设接口实质上是一个将串行传输数据位发送到其它 SPI 的移位寄存器。数据传输时，一个 SPI 系统作为 " 主机 " 控制数据流，其它 SPI 作为 " 从机 " ，主机控制数据的移入与移出。不同的 CPU 可轮流作为主机 (多主机协议与单主机协议不同，单主机协议中只有一个 CPU 始终作为主机，其它 CPU 始终作为从机) 且一个主机可同时将数据移入多从机。但只允许单从机将其数据写入主机。

当主机插入 NSS 信号时，选定一个从机。若有多从机存在，主机对每个从机产生一个独立的从机选择信号 (NPCS)。

SPI 系统包括两条数据线及两条控制线：

- 主机输出从机输入 (MOSI)：该数据线将主机输出数据作为从机输入移入。
- 主机输入从机输出 (MISO)：该数据线将从机输出作为主机输入。传输时，只有单从机传输数据。
- 串行时钟 (SPCK)：该控制线由主机驱动，用来调节数据流。主机传输数据波特率可变；每传输一位，产生一个 SPCK 周期。
- 从机选择 (NSS)：该控制线允许硬件开关从机。

SPI 主要特性如下：

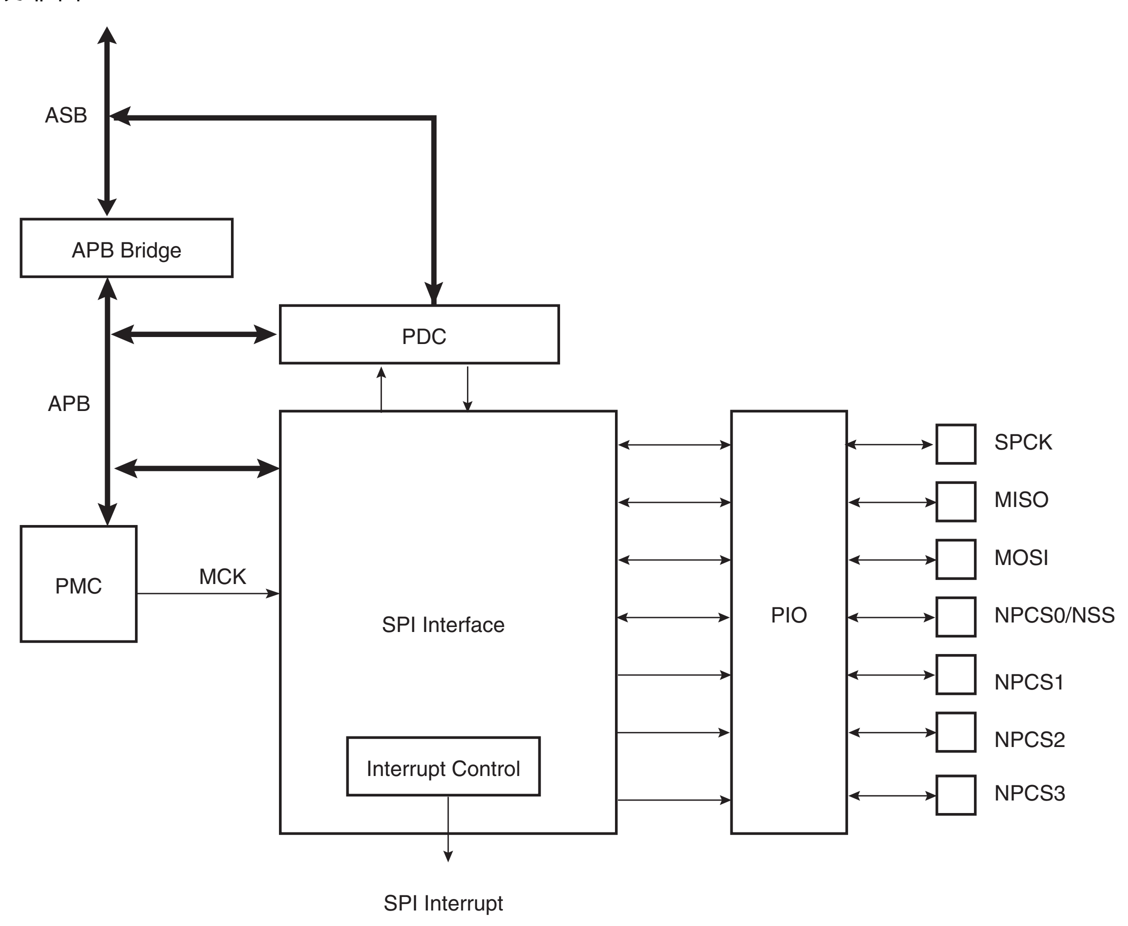
- 支持与串行外设器件通信
  – 有外部解码器的 4 个片选，最多可与 15 个外设通信
  – 串行存储器，如 DataFlash 及 3 线 EEPROM
  – 串行外设，如 ADC、 DAC、 LCD 控制器、 CAN 控制器及传感器
  – 外部协处理器
- 主机或从机串行外设总线接口
  – 每个片选线， 8 到 16 位可编程数据长
  – 每个片选线，可编程相位与极性
  – 每个片选线，连续传输时或时钟与数据间的可编程发送延迟
  – 连续传输间可编程延迟
  – 可选错误模式检测
- 与 PDC 通道连接以优化数据传输
  – 发送器与接收器各一个通道
  – 支持下一级缓冲器





方框图

Figure 153. 方框图



应用框图

Figure 154. 应用框图：单主机 / 多从机

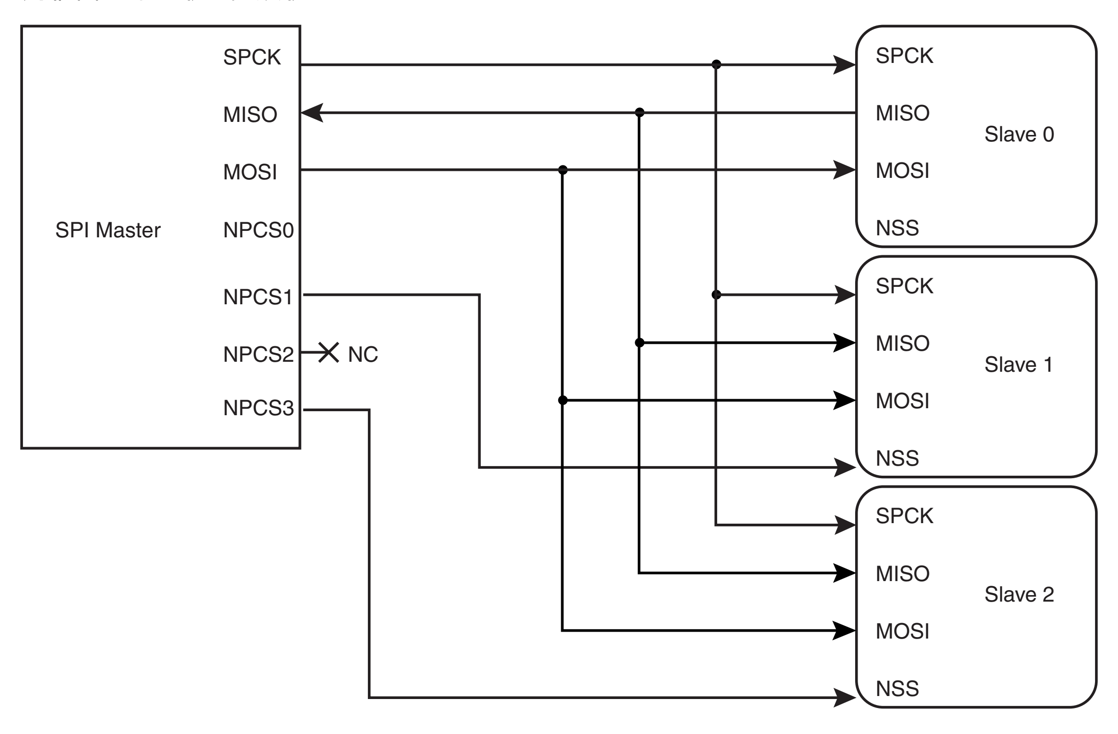


Table 68. 信号说明

引脚名称	引脚说明	类型	
		主机	从机
MISO	主入从出	输入	输出
MOSI	主出从入	输出	输入
SPCK	串行时钟	输出	输入
NPCS1-NPCS3	外设片选	输出	未用
NPCS0/NSS	外设片选 / 从机选择	输出	输入

附属产品

I/O 线

该引脚用来连接适用的外设，可与 PIO 线复用。程序员要先对 PIO 控制器编程，将 SPI 引脚分配给外设。

电源管理

SPI 可由电源管理控制器 (PMC) 提供时钟，因此，程序员必须先配置 PMC 以使能 SPI 时钟。

中断

SPI接口有与高级中断控制器(AIC)连接的中断线。处理SPI中断请求须在配置SPI前对AIC编程。

功能描述

工作模式

当配置为主机模式时，串行外设接口控制与 SPI 总线连接的从机的数据传输。 SPI 驱动片选信号到给从机或驱动串行时钟 (SPCK)。使能 SPI 后，一旦内核写入 SPI_TDR (发送数据寄存器)，立即开始数据传输。

发送与接收缓冲器的数据流维持在一个稳定的速度上，可减少高优先级中断请求。当 SPI_TDR 中新数据有效 ,SPI 继续发送数据。若 SPI_RDR (接收数据寄存器) 在收到新数据前未被读取，溢出错误 (OVRES) 标志置位。

Note:一旦该标志置位， SPI_RDR 中将不会载入数据。用户须读状态寄存器将标志清除。

片选有效与数据传输 (DLYBS) 间可编程延迟与数据发送 (DLYBCT) 间延迟，均可由四个外设片选中的各个片选编程。所有数据传输特性，包括两定时值均在 SPI_CSR0 到 SPI_CSR3 (片选寄存器) 中设定。

主机模式下，外设选择可定义为两种方式：

- 固定外设选择：SPI 仅与一个外设交换数据
- 可变外设选择：数据可与多个外设交换数据

Figure 159 与 Figure 160 给出 SPI 在主机模式下的操作方式。关于该模块中的标志与控制位的更多细节见程序模式中的表。

固定外设选择

该模式用来传输存储块，且不需要发送数据寄存器用来确定外设的额外开销。

通过在SPI_MR (模式寄存器)的PS位写0来激活固定外设选择。外设由SPI_MR的PCS域定义。

该选项仅在主机模式下 SPI 编程有效。

可变外设选择

将 PS 设置为 1，激活可变外设选择。 SPI_TDR 中的 PCS 域用来选择目标外设。当选定外设改变时，数据传输特性根据相关片选寄存器改变。

SPI_MR 中的 PCS 域无效。

该选项仅在主机模式下 SPI 编程有效。

片选

主机模式下，片选线由 SPI 驱动。这些线用来选择目标外设。SPI_MR 中的 PCSDEC 域选择 1 到 4 个外设 (PCSDEC = 0) 或高达 15 个外设 (PCSDEC = 1)。

若可变外设选择激活，所有传输的片选信号定义在 SPI_TDR 中的 PCS 域。因此每次传输片选信号可独立定义。

若固定外设选择激活，所有传输的片选信号定义在 SPI_MR 中的 PCS 域。若需要和新的外设传输，则软件须等待当前传输结束，然后在 SPI_TDR 写入新值前改变 SPI_MR 中的 PCS 值。

每次传输结束后的 NPCS 引脚值可从 SPI_RDR (接收数据寄存器) 中得到。

默认情况下，每次传输前后。所有 NPCS 信号为高 (等于 1)。

时钟产生与传输延迟

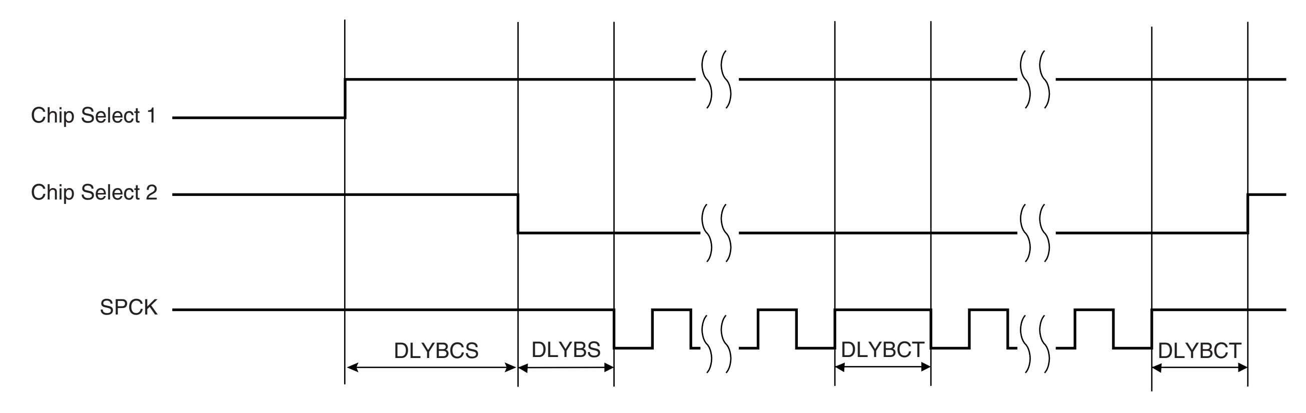
SPI波特率时钟由主机时钟(MCK)分频或主机时钟32 (若模式寄存器的DIV32设置) 与4 或510乘积值分频。分频器在每个片选寄存器的 SCBR 域中定义。因此对于每个片选信号的传输速度可独立定义。

Figure 155 给出片选传输改变及相同片选中的连续传输。有三种延迟可编程以修改传输波形 :

- 片选间延迟，对于所有片选只可通过写模式寄存器的 DLYBCS 域改变一次。允许在释放芯片及开始新传输前插入一个延时。
- SPCK 前延迟 , 通过写 DLYBS 域实现 , 对每个片选独立可编程。允许在片选出现后 SPCK 启动延迟。
- 连续传输延迟，通过写 DLYBCT 域实现，对每个片选独立可编程。在同一芯片两传输间插入一个延迟。

这些延迟允许 SPI 调整与外设连接及它们的速度及总线释放时间。

Figure 155. 可编程延迟



模式错误检测

当 SPI 编程为主机模式且外部主机将 NPCS0/NSS 信号驱动为低电平时，检测到模式错误。

当检测到模式错误， MODF 位在读 SPI_SR 前置位，而 SPI 自动禁用直到在 SPI_CR(控制寄存器) 的 SPIEN 位写 1 将其重新使能为止。

默认情况下，模式错误检测电路使能。用户可通过设置 SPI_MR 中的 MODFDIS 位禁用模式错误检测。

主机模式流程图

Figure 156. 主机模式流程图

SPI Enable

TDRE

1

0

PS

0 Fixed peripheral

1 Variable peripheral

NPCS = SPI_TDR(PCS)

NPCS = SPI_MR(PCS)

Delay DLYBS

Serializer = SPI_TDR(TD)
TDRE = 1

Data Transfer

SPI_RDR(RD) = Serializer
RDRF = 1

Delay DLYBCT

TDRE

0

1

NPCS = 0xF

Delay DLYBCS

PS

0 Fixed peripheral

1 Variable peripheral

SPI_TDR(PCS)

Same peripheral

New peripheral

NPCS = 0xF

Delay DLYBCS

NPCS = SPI_TDR(PCS)

主机模式框图

Figure 157. 主机模式框图

SPI_MR(DIV32)
MCK
MCK/32
0
1
SPCK Clock Generator
SPI_CSRx[15:0]
SPCK
SPIDIS
SPIEN
S
Q
R
SPI_RDR
PCS
RD
LSB
MSB
MISO
Serializer
MOSI
SPI_TDR
PCS
TD
NPCS3
NPCS2
NPCS1
NPCS0
SPI_MR(PS)
1
0
SPI_MR(PCS)
SPI_MR(MSTR)
SPI_SR
MODF
TDRE
RDRF
OVRE
SPIENS
SPI_IER
SPI_IDR
SPI_IMR
SPI Interrupt

SPI 从机模式

从机模式下，SPI 在由外部主机接收串行时钟前等待 NSS 激活。

从机模式下，SPI_CSR0 中的 CPOL、NCPHA 与 BITS 域用来定义发送特性。其它片选寄存器在从机模式下未用。

从机模式下，SPCK 上输入时钟由低到高的脉冲持续时间必须长于两个主机时钟周期。

Figure 158. 从机模式框图

数据传输

数据传输有四种极性与相位。时钟极性由片选寄存器 CPOL 位编程得到。时钟相位由 NCPHA 位编程得到。这两个参数确定数据在哪个时钟边沿驱动与采样。每个参数有两种状态，组合后



有四种可能。因此，一对主机 / 从机必须使用相同的参数对值来进行通信。若使用多从机，且固定为不同的配置，主机与不同从机通信时必须重新配置。

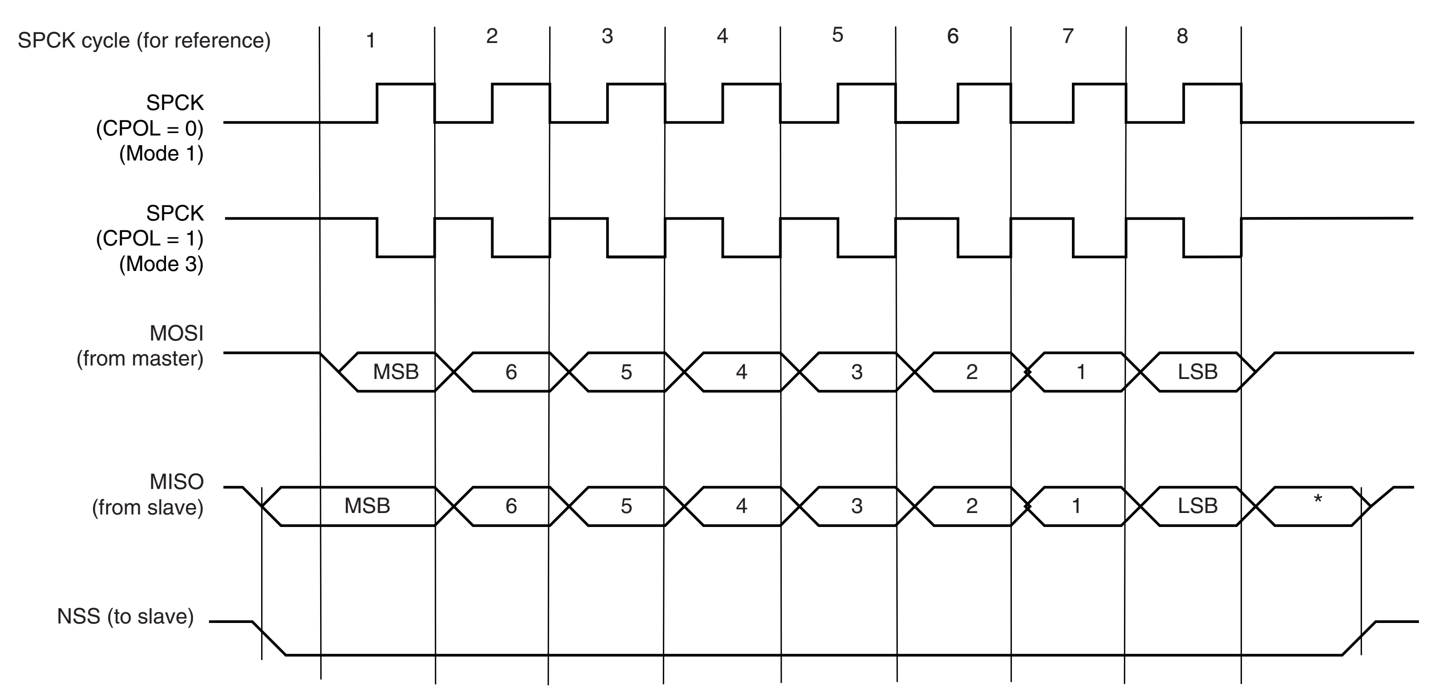
Table 69 给出四种模式与相应的参数设置。

Table 69. SPI 总线协议模式

SPI 模式	CPOL	NCPHA
0	0	0
1	0	1
2	1	0
3	1	1

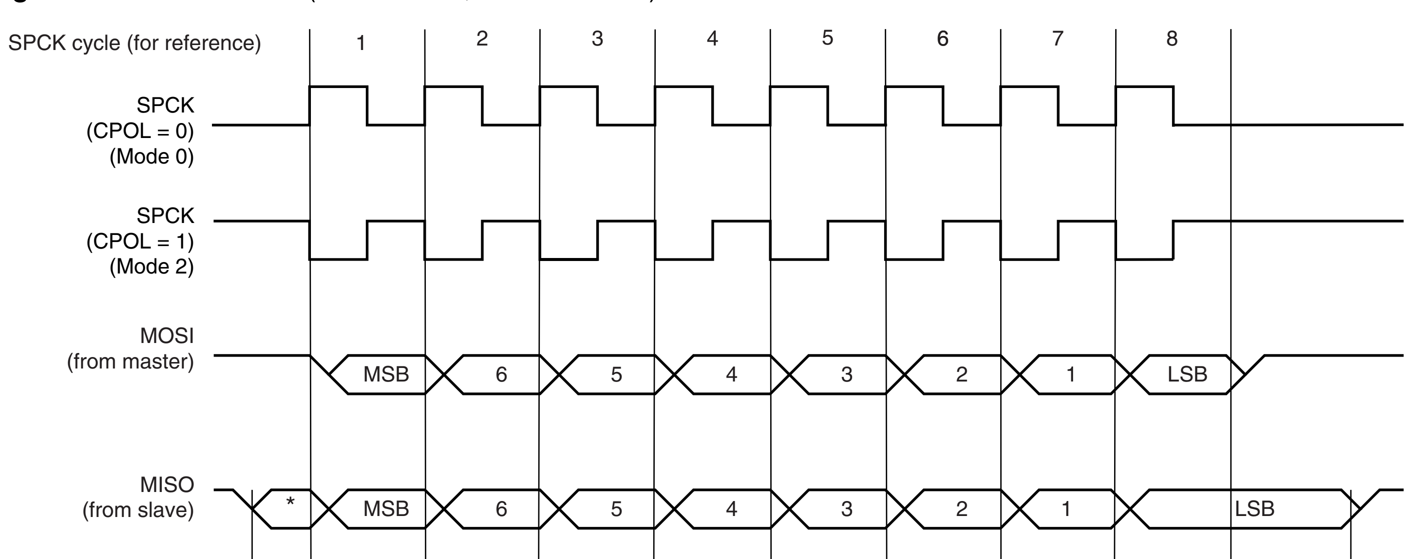
Figure 159 与 Figure 160 给出数据传输示例。

Figure 159. SPI 传输格式 (NCPHA = 1，每次传输 8 位)



* Not defined, but normally MSB of previous character received.

Figure 160. SPI 传输格式 (NCPHA = 0，每次传输 8 位)



* Not defined but normally LSB of previous character transmitted.



串行外设接口 (SPI) 用户接口

Table 70. SPI 寄存器映射

偏移	寄存器	寄存器名称	访问类型	复位值
0x00	控制寄存器	SPI_CR	只写	---
0x04	模式寄存器	SPI_MR	读 / 写	0x0
0x08	接收数据寄存器	SPI_RDR	只读	0x0
0x0C	发送数据寄存器	SPI_TDR	只写	---
0x10	状态寄存器	SPI_SR	只读	0x000000F0
0x14	中断使能寄存器	SPI_IER	只写	---
0x18	中断禁用寄存器	SPI_IDR	只写	---
0x1C	中断屏蔽寄存器	SPI_IMR	只读	0x0
0x20 - 0x2C	保留			
0x30	片选寄存器 0	SPI_CSR0	读 / 写	0x0
0x34	片选寄存器 1	SPI_CSR1	读 / 写	0x0
0x38	片选寄存器 2	SPI_CSR2	读 / 写	0x0
0x3C	片选寄存器 3	SPI_CSR3	读 / 写	0x0
0x40 - 0xFF	保留	–	–	
0x100 - 0x124	保留给 PDC			

SPI 控制寄存器

寄存器名称 : SPI_CR

访问类型 : 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
SWRST	–	–	–	–	–	SPIDIS	SPIEN

- **SPIEN: SPI 使能**

0 = 无效。

1 = 使能 SPI 发送与接收数据。

- **SPIDIS: SPI 禁用**

0 = 无效。

1 = 禁用 SPI。

所有引脚设置为输入模式，无数据发送或接收。

若正处理传输，传输结束后禁用 SPI。

当控制寄存器写入时若 SPIEN 与 SPIDIS 均为 1， SPI 禁用。

- **SWRST: SPI 软件复位**

0 = 无效。

1 = 复位 SPI。

执行 SPI 接口的软件触发硬件复位。



SPI 模式寄存器

寄存器名称： SPI_MR

访问类型： 读 / 写

31	30	29	28	27	26	25	24
DLYBCS							

23	22	21	20	19	18	17	16
–	–	–	–	PCS			

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
LLB	–	–	MODFDIS	DIV32	PCSDEC	PS	MSTR

- **MSTR: 主机 / 从机模式**

0 = SPI 为从机模式。

1 = SPI 为主机模式。

- **PS: 外设选择**

0 = 固定外设选择。

1 = 可变外设选择。

- **PCSDEC: 片选译码**

0 = 片选直接与外设连接

1 = 四个片选与 4 到 16 位译码器连接。

当 PCSDEC 等于 1，使用外部 4 到 16 位译码器可产生 16 个片选信号。

片选寄存器通过下列规则定义 16 个片选特性：

SPI_CSR0 定义外设片选信号 0 到 3。

SPI_CSR1 定义外设片选信号 4 到 7。

SPI_CSR2 定义外设片选信号 8 到 11。

SPI_CSR3 定义外设片选信号 12 到 15*。

***Note**: 第 16 个状态对应于将所有片选失效。允许对每个片选寄存器独立配置时钟。*

- **FDIV: 时钟选择**

0 = SPI 时钟为 MCK。

1 = SPI 时钟为 MCK/32。

- **MODFDIS: 模式错误检测**

0 = 模式错误检测使能。

1 = 模式错误检测禁用。

- **LLB: 本地回环使能**

0 = 本地回环路径禁用。

1 = 本地回环路径使能。

LLB 在主机模式下控制数据串行器的本地回环。

- **PCS: 外设片选**

该域仅适用于固定外设选择有效 (PS = 0)。

若 PCSDEC = 0：

PCS = xxx0　　NPCS[3:0] = 1110

PCS = xx01 NPCS[3:0] = 1101

PCS = x011 NPCS[3:0] = 1011

PCS = 0111 NPCS[3:0] = 0111

PCS = 1111 禁用 (未选择外设)

(x = 不必在意)

若 PCSDEC = 1 :

NPCS[3:0] 输出信号 = PCS。

- **DLYBCS: 片选延迟**

该域定义 NPCS 无效到其它 NPCS 有效间的延迟。DLYBCS 时间保证片选不重叠并解决外设长时间数据流引起的总线竞争。

若 DLYBCS 小于或等于 6 , 6 个 MCK 周期 (若 DIV32 置位 , 192 MCK 周期) 将缺省插入。

其它情况下 , 按照下列等式确定延迟 :

若 DIV32 为 0 :

$$\text{Delay Between Chip Selects} = \text{DLYBCS} / \text{MCK}$$

若 DIV32 为 1 :

$$\text{Delay Between Chip Selects} = \text{DLYBCS} \times 32 / \text{MCK}$$



SPI 接收数据寄存器

寄存器名称： SPI_RDR

访问类型： 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	PCS			
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RD							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RD							

- **RD: 接收数据**

SPI 接口收到的数据存于该寄存器的相应位中。未用位值为 0。

- **PCS: 外设片选**

仅在主机模式下，这些位表示传输结束后 NPCS 引脚值；其它情况下，这些位值为 0。

SPI 发送数据寄存器

寄存器名称 : SPI_TDR

访问类型 : 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	PCS			

15	14	13	12	11	10	9	8
TD							

7	6	5	4	3	2	1	0
TD							

- **TD: 发送数据**

SPI 接口发送数据存于该寄存器中。要发送的信息必须以正确的格式写入数据发送寄存器中。

PCS: 外设片选

该域仅适用于可变外设选择有效 (PS = 1)。

若 PCSDEC = 0 :

PCS = xxx0 NPCS[3:0] = 1110
PCS = xx01 NPCS[3:0] = 1101
PCS = x011 NPCS[3:0] = 1011
PCS = 0111 NPCS[3:0] = 0111
PCS = 1111 禁用 (未选择外设)
(x = 不必在意)

若 PCSDEC = 1 :

NPCS[3:0] 输出信号 = PCS



SPI 状态寄存器

寄存器名称： SPI_SR

访问类型： 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	SPIENS
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TXBUFE	RXBUFF	ENDTX	ENDRX	OVRES	MODF	TDRE	RDRF

- **RDRF: 接收数据寄存器满**

0 = 上次读 SPI_RDR 后未收到数据。

1 = 上次读 SPI_RDR 后已收到数据并由串行器发送到 SPI_RDR。

- **TDRE: 发送数据寄存器空**

0 = 数据已写入 SPI_TDR 但仍未传输到串行器。

1 = 最后写入发送数据寄存器的数据已传输到串行器。

当 SPI 禁用或复位时， TDRE 等于 0 。 SPI 使能命令设置该位为 1。

- **MODF: 模式错误检测**

0 = 上次读 SPI_SR 后未检测到模式错误。

1 = 上次读 SPI_SR 后出现模式错误。

- **OVRES: 溢出错误状态**

0 = 上次读 SPI_SR 后未检测到溢出错误。

1 = 上次读 SPI_SR 后出现溢出错误。

上次读 SPI_RDR 后当 SPI_RDR 至少两次载入串行器，出现溢出错误。

- **ENDRX: RX 缓冲结束**

0 = 上次写 SPI_RCR 或 SPI_RNCR 后接收计数寄存器仍未达到 0。

1 = 上次写 SPI_RCR 或 SPI_RNCR 后接收计数寄存器已达到 0。

- **ENDTX: TX 缓冲结束**

0 = 上次写 SPI_TCR 或 SPI_TNCR 后发送计数寄存器仍未达到 0。

1 = 上次写 SPI_TCR 或 SPI_TNCR 后发送计数寄存器已达到 0。

0 = SPI_RCR 或 SPI_RNCR 是非 0 值。

1 = SPI_RCR 与 SPI_RNCR 值为 0。

- **TXBUFE: TX 缓冲器空**

0 = SPI_TCR 或 SPI_TNCR 是非 0 值。

1 = SPI_TCR 与 SPI_TNCR 值为 0。

- **SPIENS: SPI 使能状态**

0 = SPI 禁用。

1 = SPI 使能。

SPI 中断使能寄存器

寄存器名称 : SPI_IER

访问类型 : 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TXBUFE	RXBUFF	ENDTX	ENDRX	OVRES	MODF	TDRE	RDRF

- **RDRF: 接收数据寄存器满中断使能**
- **TDRE: SPI 发送数据寄存器空中断使能**
- **MODF: 模式错误检测中断使能**
- **OVRES: 溢出错误中断使能**
- **ENDRX: 接收缓冲器结束中断使能**
- **ENDTX: 发送缓冲器结束中断使能**
- **RXBUFF: 接收缓冲器满中断使能**
- **TXBUFE: 发送缓冲器空中断使能**

0 = 无效。

1 = 使能相应中断。

SPI 中断禁用寄存器

寄存器名称 : SPI_IDR

访问类型 : 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
TXBUFE	RXBUFF	ENDTX	ENDRX	OVRES	MODF	TDRE	RDRF

- **RDRF: 接收数据寄存器满中断禁用**
- **TDRE: SPI 发送数据寄存器空中断禁用**
- **MODF: 模式错误检测中断禁用**
- **OVRES: 溢出错误中断禁用**
- **ENDRX: 接收缓冲器结束中断禁用**
- **ENDTX: 发送缓冲器结束中断禁用**
- **RXBUFF: 接收缓冲器满中断禁用**
- **TXBUFE: 发送缓冲器空中断禁用**

0 = 无效。

1 = 禁用相应中断。

SPI 中断屏蔽寄存器

寄存器名称 : SPI_IMR

访问类型 : 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
TXBUFE	RXBUFF	ENDTX	ENDRX	OVRES	MODF	TDRE	RDRF

- **RDRF: 接收数据寄存器满中断屏蔽**
- **TDRE: SPI 发送数据寄存器空中断屏蔽**
- **MODF: 模式错误检测中断屏蔽**
- **OVRES: 溢出错误中断屏蔽**
- **ENDRX: 接收缓冲器结束中断屏蔽**
- **ENDTX: 发送缓冲器结束中断屏蔽**
- **RXBUFF: 接收缓冲器满中断屏蔽**
- **TXBUFE: 发送缓冲器空中断屏蔽**

0 = 相应中断未使能。

1 = 相应中断使能。



SPI 片选寄存器

寄存器名称： SPI_CSR0... SPI_CSR3

访问类型： 读 / 写

31	30	29	28	27	26	25	24
DLYBCT							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
DLYBS							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
SCBR							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
BITS				–	–	NCPHA	CPOL

- **CPOL: 时钟极性**

0 = SPCK 无效状态值为逻辑 0。

1 = SPCK 无效状态值为逻辑 1。

CPOL 用来确定串行时钟 (SPCK) 的无效状态值，它和 NCPHA 一起使用产生主机与从机间所需的时钟 / 数据关系。

- **NCPHA: 时钟相位**

0 = 数据在 SPCK 起始边沿改变，在 SPCK 下一个边沿捕获。

1 = 数据在 SPCK 起始边沿捕获，在 SPCK 下一个边沿改变。

NCPHA 确定 SPCK 的哪个边沿引起数据改变，哪个边沿引起数据捕获。NCPHA 与 CPOL 一起使用产生主机与从机间所需的时钟 / 数据关系。

- **BITS: 传输位**

BITS 域确定传输数据位数，不使用保留值。

BITS[3:0]	传输位
0000	8
0001	9
0010	10
0011	11
0100	12
0101	13
0110	14
0111	15
1000	16
1001	保留
1010	保留
1011	保留
1100	保留
1101	保留
1110	保留
1111	保留

- **SCBR: 串行时钟波特率**

主机模式下， SPI 接口使用模块接收器由主机时钟 MCK 中导出 SPCK 波特率。可在 SCBR 域中写入 2 到 255 间的值来选择波特率。下列等式确定 SPCK 波特率：

若 DIV32 为 0：

SPCK Baudrate ＝ MCK／（2 × SCBR）

若 DIV32 为 1：

SPCK Baudrate ＝ MCK／（64 × SCBR）

给定 SCBR 值为 0 或一个禁用波特率发生器。 SPCK 禁用并恢复到无效状态值。不出现串行传输。复位时，波特率禁用。

- **DLYBS: SPCK 前延时**

该域定义由 NPCS 有效到第一次有效 SPCK 传输间的延迟。

当 DLYBS 等于 0， NPCS 有效到 SPCK 传输为 1/2 个 SPCK 时钟周期。

其它情况下，由下列等式决定延迟。

若 DIV32 为 0：

Delay Before SPCK ＝ DLYBS／ MCK

若 DIV32 为 1：

Delay Before SPCK ＝ 32 × DLYBS／ MCK

- **DLYBCT: 连续传输间延迟**

该域定义无片选删除时相同外设的两次传输间的延迟。若需要的话，该延迟在每次传输后删除片选前插入。

当 DLYBCT 等于 0，连续传输中不插入延迟，且时钟保持其占空覆盖字符传输。

其它情况下，由下列等式决定延迟。

若 DIV32 为 0：

Delay Between Consecutive Transfers ＝ 32 × DLYBCT／ MCK

若 DIV32 为 1：

Delay Between Consecutive Transfers ＝ 1024 × DLYBCT／ MCK



两线接口 (TWI)

概述

两线接口 (TWI) 由一根时钟线及一根传输速度达到 400 Kb/s 的数据线组成，以字节为单位进行传输。它适用于任何的 Atmel 两线总线串行 EEPROM 中。 TWI 可编程作为主机进行连续或单字节访问。

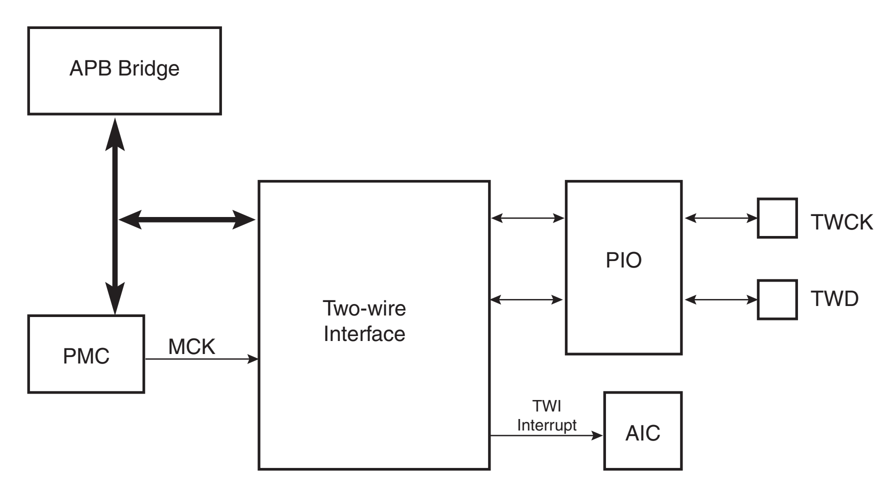
可配置波特率发生器允许输出数据速率在内核时钟频率的一个宽范围内进行调整。

TWI 主要特性如下：

- 与标准两线串行存储器兼容
- 1、 2、 3 字节从机地址
- 连续读 / 写操作

方框图

Figure 161. 方框图



应用框图

Figure 162. 应用框图

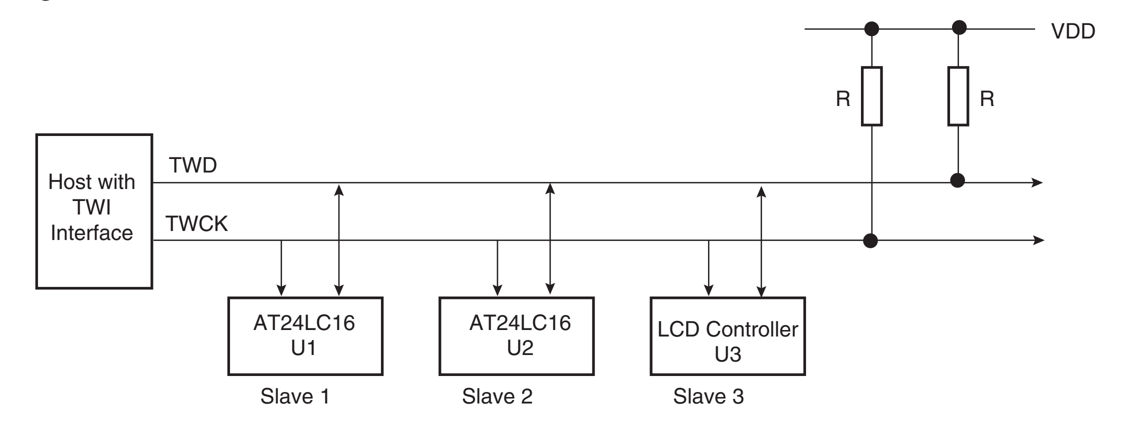




Table 71. I/O 线说明

引脚名称	引脚说明	类型
TWD	两线串行数据	输入 / 输出
TWCK	两线串行时钟	输入 / 输出

附属产品

I/O 线

TWD与TWCK为双向线，通过当前源正极或上拉电阻连接(见Figure 162 on page 377)。当总线空闲时，两线均为高。连接到总线上的输出流水线必须有一个开漏或开集来执行线与功能。

TWD 与 TWCK 引脚可与 PIO 线复用。为使能 TWI，必须执行下列步骤：

- 将 PIO 控制器编程为：
  - 将 TWD 与 TWCK 指定为外设线。
  - 将 TWD 与 TWCK 定义为开漏。

电源管理

- 使能外设时钟。

TWI 接口可通过电源管理控制器 (PMC) 提供时钟，因此必须先配置 PMC 以使能 TWI 时钟。

中断

TWI接口有一条与高级中断控制器(AIC)连接的中断线。为处理中断，在配置TWI前必须对AIC编程。

功能说明

传输格式

TWD线上数据必须为8位。数据传输是高位在先；每字节后必须有应答信号。每次传输的字节数目没有限制 (见 Figure 164 on page 379)。

每次传输以 START 状态开始，以 STOP 状态停止 (见 Figure 163 on page 378)。

- 当 TWCK 为高时 TWD 由高变低定义为 START 状态。
- 当 TWCK 为高时 TWD 由低变高定义为 STOP 状态。

Figure 163. START 与 STOP 状态

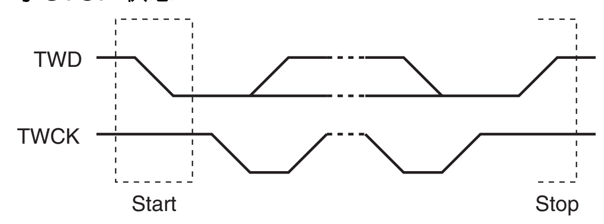
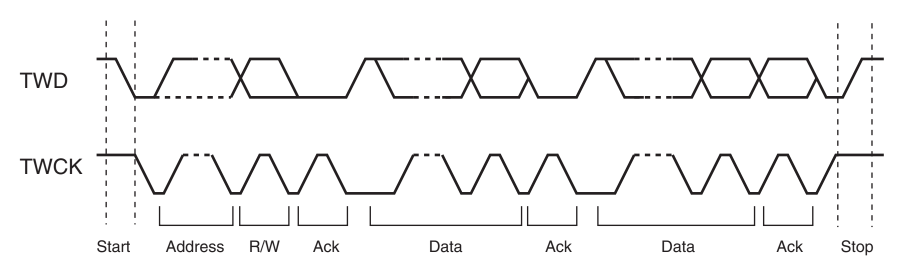


Figure 164. 传输格式



工作模式

TWI 有两种工作模式：

- 主机发送模式
- 主机接收模式

主机模式下， TWI 控制寄存器 (TWI_CR) 可配置为接口。该模式下，根据在时钟波形发生器寄存器 (TWI_CWGR) 中编程值产生时钟。该寄存器定义定义了 TWCK 信号，使能接口以适应宽范围时钟。

数据发送

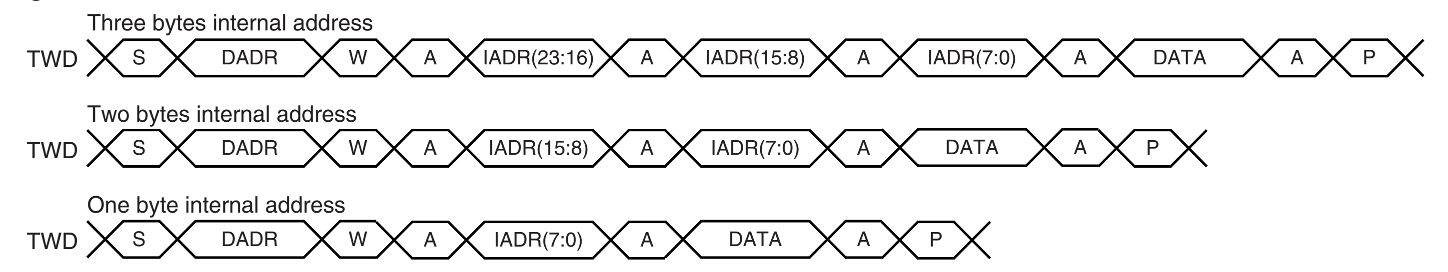
主机初始化 Start 状态后 ,向主机模式寄存器 (TWI_MMR 中 DADR) 发送一个 7 位从机地址 ,以通知从机器件。从机地址后的位表示传输方向 (写或读)。该位为 0，说明是写操作 (发送操作)；若该位为 1，说明为数据读请求 (接收操作)。

TWI 传输要求从机每收到一个字节后均要给出应答。在应答时钟脉冲中，主机释放数据线 (HIGH)，将从机拉低以产生应答。主机在该时钟脉冲中轮询数据线，若从机未应答该字节将置位状态寄存器的 **NAK** 位。与其它状态位相同 , 若使能中断使能寄存器 (TWI_IER) 将产生中断。写发送保持寄存器 (TWI_THR) 后，设置控制寄存器的 START 位以启动传输。数据在内部移位寄存器中移位，当检测到应答，TXRDY 位置位，直到 TWI_THR 中有新数据写入，才清除该位 (见 Figure 166 on page 380)。主机产生 STOP 状态来结束传输。

设置 START 后开始读序列。当状态寄存器中 RXRDY 位置位时，接收保持寄存器 (TWI_RHR) 以收到一个字符。当读 TWI_RHR 时 RXRDY 位复位。

TWI接口可执行多种传输格式 (7位从机地址 ,10位从机地址)。通过主机模式寄存器(TWI_MMR)配置三个内部地址字节。若从机仅支持 7 位地址，**IADRSZ** 必须置为 0。若从机地址大于 7 位，用户必须配置地址大小 **(IADRSZ)** 并在内部地址寄存器 (TWI_IADR) 中设置其它从机地址位。

Figure 165. 1、 2 或 3 字节内部地址及 1 字节数据的主机写操作



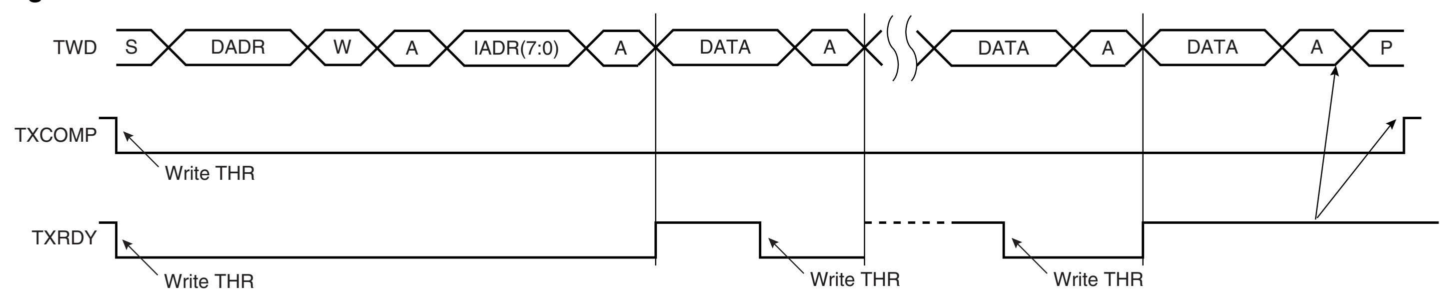


Figure 166. 1 字节内部地址及多数据字节的主机写操作

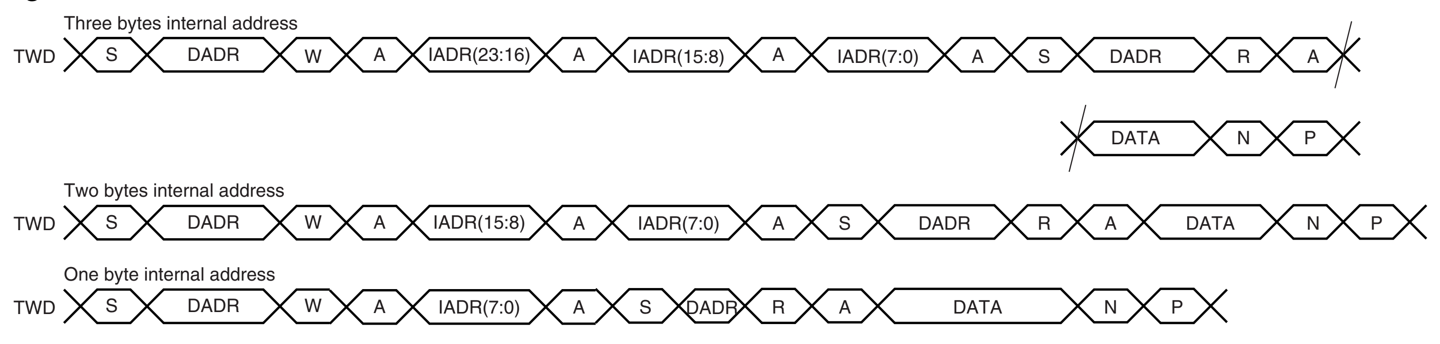


Figure 167. 1、 2 或 3 字节内部地址及 1 字节数据的主机读操作

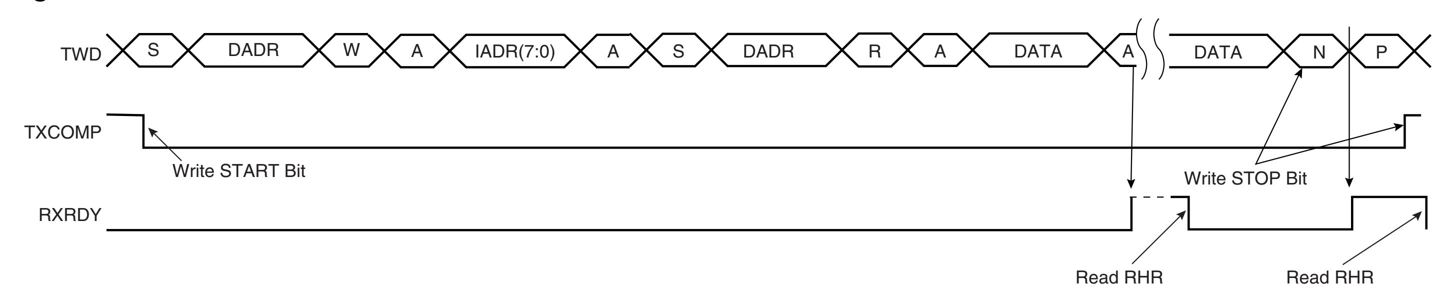
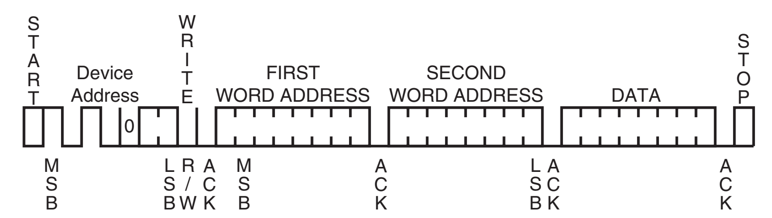


Figure 168. 1 字节内部地址及多数据字节的主机读操作

- S = Start
- P = Stop
- W = 读 / 写
- A = 应答
- DADR= 器件地址
- IADR = 内部地址

Figure 169给出了在Atmel AT24LC512 EEPROM中写入一字节的图示。该图示范了如何使用内部地址访问器件。

Figure 169. 内部地址用法



读 / 写流程图

Figure 170 on page 382 与 Figure 171 on page 383 中流程图给出主机模式下读、写操作示例。可使用轮询或中断方式来检验状态位。使用中断方式时要先配置中断使能寄存器 (TWI_IER)。



Figure 170. 主机模式下 TWI 写操作

START

Set TWI clock:
TWI_CWGR = clock

Set the control register:
- Master enable
TWI_CR = TWI_MSEN

Set the Master Mode register:
- Device slave address
- Internal address size
- Transfer direction bit
Write ==> bit MREAD = 0

Internal address size = 0?

Set the internal address
TWI_IADR = address

Yes

Load transmit register
TWI_THR = Data to send
Start the transfer
TWI_CR = TWI_START

Read status register

TWI_THR = data to send

TXRDY = 0?

Yes

Data to send?

Yes

Stop the transfer
TWI_CR = TWI_STOP

Read status register

TXCOMP = 0?

END

Figure 171. 主机模式下 TWI 读操作

START

Set TWI clock:
TWI_CWGR = clock

Set the control register:
- Master enable
TWI_CR = TWI_MSEN

Set the Master Mode register:
- Device slave address
- Internal address size
- Transfer direction bit
Read ==> bit MREAD = 0

Internal address size = 0?

Set the internal address
TWI_IADR = address

Yes

Start the transfer
TWI_CR = TWI_START

Read status register

RXRDY = 0?

Yes

Data to read?

Yes

Stop the transfer
TWI_CR = TWI_STOP

Read status register

TXCOMP = 0?

Yes

END

两线接口 (TWI) 用户接口

Table 72. TWI 寄存器映射

偏移	寄存器	名称	访问类型	复位值
0x0000	控制寄存器	TWI_CR	只写	N/A
0x0004	主机模式寄存器	TWI_MMR	读 / 写	0x0000
0x0008	保留		–	
0x000C	内部地址寄存器	TWI_IADR	读 / 写	0x0000
0x0010	时钟波形发生寄存器	TWI_CWGR	读 / 写	0x0000
0x0020	状态寄存器	TWI_SR	只读	0x0008
0x0024	中断使能寄存器	TWI_IER	只写	N/A
0x0028	中断禁用寄存器	TWI_IDR	只写	N/A
0x002C	中断屏蔽寄存器	TWI_IMR	只读	0x0000
0x0030	接收保持寄存器	TWI_RHR	只读	0x0000
0x0034	发送保持寄存器	TWI_THR	读 / 写	0x0000

TWI 控制寄存器

寄存器名称： TWI_CR

访问类型： 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SWRST	–	–	–	MSDIS	MSEN	STOP	START

- **START: 发送 START 状态**

0 = 无效。

1 = 根据模式寄存器定义的特性，帧传输是以发送 START 位开始的。

当 TWI 外设需要由从机读数据时该步骤必须执行。当在主机模式下配置一个写操作，用户在保持寄存器写入一个字符，则使用模式寄存器开始发送一帧数据。

- **STOP: 发送 STOP 状态**

0 = 无效。

1 = 主机读或写模式下，当前字节传输完成后，发送 STOP 状态。

单数据字节主机读写时， START 与 STOP 必须设置。

多数据字节主机读写时， STOP 在传输 ACK/NACK 位前必须设置。

主机读模式下，若收到 NACK 位， STOP 自动执行。

多数据字节写操作时，当 THR 与移位寄存器均为空时，自动发送 STOP 状态。

- **MSEN: TWI 主机传输使能**

0 = 无效。

1 = 若 MSDIS = 0，主机数据传输使能。

- **MSDIS: TWI 主机传输禁用**

0 = 无效。

1 = 主机数据传输禁用，发送所有挂起数据。写操作时，发送移位和保持字符 (若其中包含数据)；读操作时，在禁用前接收所有传输字符。

- **SWRST: 软件复位**

0 = 无效。

1 = 与系统复位等效。



TWI 主机模式寄存器

寄存器名称： TWI_MMR

访问类型： 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	DADR						
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	MREAD	–	–	IADRSZ	
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	–	–	–	–	–	–

- **IADRSZ: 内部器件地址长**

IADRSZ[9:8]		
0	0	无内部器件地址
0	1	1 字节内部器件地址
1	0	2 字节内部器件地址
1	1	3 字节内部器件地址

- **MREAD: 主机读方向**

0 = 主机写。

1 = 主机读。

- **DADR: 器件地址**

器件地址用于在主机模式读或写下访问从机器件。

TWI 内部地址寄存器

寄存器名称： TWI_IADR

访问类型： 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
IADR							
15	14	13	12	11	10	9	8
IADR							
7	6	5	4	3	2	1	0
IADR							

- **IADR:** 内部地址

0、 1、 2 或 3 字节由 IADRSZ 决定。

TWI 时钟波形发生器寄存器

寄存器名称： TWI_CWGR

访问类型： 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	CKDIV		
15	14	13	12	11	10	9	8
CHDIV							
7	6	5	4	3	2	1	0
CLDIV							

- **CLDIV:** 时钟低分频器

TWCK 低周期定义如下：

$$T_{low} = ((\mathrm{CLDIV} \times 2^{\mathrm{CKDIV}}) + 3) \times T_{MCK}$$

- **CHDIV:** 时钟高分频器

TWCK 高周期定义如下：

$$T_{high} = ((\mathrm{CHDIV} \times 2^{\mathrm{CKDIV}}) + 3) \times T_{MCK}$$

- **CKDIV:** 时钟分频器

TWCK 高低周期中 CKDIV 均增加。

TWI 状态寄存器

寄存器名称： TWI_SR

访问类型： 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	NACK

7	6	5	4	3	2	1	0
UNRE	OVRE	–	–	–	TXRDY	RXRDY	TXCOMP

- **TXCOMP: 传输完成**

0 = 对于主机，为当前帧长；对于从机，为从收到 START 到收到 STOP。

1 = 当保持与移位寄存器均为空且 STOP 状态已发送 (主机) 或已收到 (从机)，或 MSEN 置位 (使能 TWI)。

- **RXRDY: 接收保持寄存器就绪**

0 = 上次 TWI_RHR 读操作后未收到字符。

1 = 上次 TWI_RHR 读操作后收到一字符。

- **TXRDY: 发送保持寄存器就绪**

0 = 发送保持寄存器中内容未传输到移位寄存器中，当对 TWI_THR 寄存器写入时设置为 0。

1 = 当数据由TWI_THR传入内部移位寄存器或检测到NACK错误 ,TXRDY与TXCOMP及NACK同时置位。当MSEN置位时，TXRDY 同样置位 (使能 TWI)。

- **OVRE: 溢出错误**

0 = RXRDY 已置位，而 TWI_RHR 未载入数据。

1 = RXRDY 置位时， TWI_RHR 已载入数据。当 TXCOMP 置位时，通过读 TWI_SR 进行复位。

- **UNRE: 空栈读出错**

0 = 无空栈读出错。

1 = 当移位寄存器加载数据时 ,TWI_THR 中无有效数据 (TXRDY 置位)。主机模式下 ,将自动产生 STOP 位。当 TXCOMP 置位时，通过读 TWI_SR 进行复位。

- **NACK: 无应答**

0 = 每个数据字节均被远端 TWI 从机正确接收。

1 = 从机未给出应答。与 TXCOMP 同时置位。读后复位。

TWI 中断使能寄存器

寄存器名称： TWI_IER

访问类型： 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	NACK

7	6	5	4	3	2	1	0
UNRE	OVRE	–	–	–	TXRDY	RXRDY	TXCOMP

- **TXCOMP:** 传输完成
- **RXRDY:** 接收保持寄存器就绪
- **TXRDY:** 发送保持寄存器就绪
- **OVRE:** 溢出错误
- **UNRE:** 空栈读错误
- **NACK:** 无应答

0 = 无效。

1 = 使能相应中断。



TWI 中断禁用寄存器

寄存器名称 :TWI_IDR

访问类型 : 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	NACK
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
UNRE	OVRE	–	–	–	TXRDY	RXRDY	TXCOMP

- **TXCOMP: 传输完成**
- **RXRDY: 接收保持寄存器就绪**
- **TXRDY: 发送保持寄存器就绪**
- **OVRE: 溢出错误**
- **UNRE: 空栈读错误**
- **NACK: 无应答**

0 = 无效。

1 = 禁用相应中断。

TWI 中断屏蔽寄存器

寄存器名称 : TWI_IMR

访问类型 : 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	NACK
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
UNRE	OVRE	–	–	–	TXRDY	RXRDY	TXCOMP

- **TXCOMP:** 传输完成
- **RXRDY:** 接收保持寄存器就绪
- **TXRDY:** 发送保持寄存器就绪
- **OVRE:** 溢出错误
- **UNRE:** 空栈读错误
- **NACK:** 无应答

0 = 相应中断禁用。

1 = 相应中断使能。



TWI 接收保持寄存器

寄存器名称 :TWI_RHR

访问类型 : 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
RXDATA							

- **RXDATA: 主机或从机接收保持数据**

TWI 发送保持寄存器

寄存器名称 : TWI_THR

访问类型 : 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
TXDATA							

- **TXDATA: 主机或从机发送保持数据**

通用同步 / 异步收发器 (USART)

概述

通用同步异步收发器 (USART) 提供一个全双工通用同步异步串行连接。数据帧格式可编程 (数据长度，奇偶校验位，停止位数) 以支持尽可能多的标准。接收器执行奇偶错误、帧错误及溢出错误检测。接收器超时使能可变长帧处理，发送器时间保障方便与慢慢速远程器件通信。接收与发送地址位也支持多点通信。

USART 有三种测试模式：远程回环、本地回环及自动回应。

USART 支持 RS485 总线提供的特殊操作模式，通过 ISO7816 T = 0 或 T = 1 智能卡插槽、红外收发器并与调制解调器连接。硬件握手通信通过 RTS 与 CTS 引脚自动管理溢出控制。

USART 支持与使能由发送器到接收器的数据传输的外设数据控制器的连接。PDC 提供没有处理器干扰的链缓冲管理。

USART 主要特性如下：

- 可编程波特率发生器
- 5 到 9 位全双工同步或异步串行通信
  – 异步模式下， 1、 1.5 或 2 位停止位；同步模式下， 1 或 2 位停止位
  – 校验发生及错误检测
  – 帧错误检测，溢出错误检测
  – MSB 或 LSB
  – 可选间断发生及检测
  – 8 或 16 倍过采样接收器频率
  – 可选硬件握手 RTS-CTS
  – 可选调制解调信号管理 DTR-DSR-DCD-RI
  – 接收器超时与发送器时间保障
  – 有地址发生与检测的可选多点模式
- 有驱动控制信号的 RS485
- 与智能卡连接的 ISO7816， T = 0 或 T = 1 协议
  – NACK 处理，有接收与反复限制的错误计数器
- IrDA 调制与解调
  – 通信速度高达 115.2 Kbps
- 测试模式
  – 远程回环、本地回环、自动回应
- 支持与两外设数据控制器通道 (PDC) 的连接
  – 提供无处理器干涉的缓冲器发送

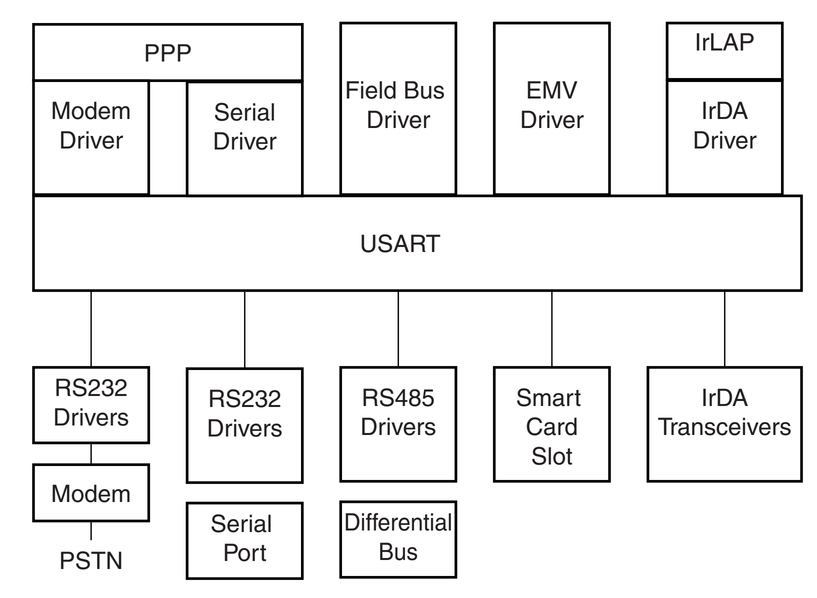


方框图

Figure 172. USART 框图

应用框图

Figure 173. 应用框图



I/O 线说明

Table 73. I/O 线说明

名称	说明	类型	有效电平
SCK	串行时钟	I/O	
TXD	发送串行数据	I/O	
RXD	接收串行数据	输入	
RI	环指示器	输入	低
DSR	数据设置就绪	输入	低
DCD	数据载波检测	输入	低
DTR	数据中止就绪	输出	低
CTS	发送清除	输入	低
RTS	发送请求	输出	低

附属产品

I/O 线

连接 USART 的引脚可与 PIO 线复用。必须先对 PIO 编程以将 USART 引脚分配到期望的外设功能。若 USART 的 I/O 线未使用， PIO 控制器可将其用作其它功能。

调制解调器的引脚不一定在 USART 中执行。通常，只有 USART1 是完全为调制解调器信号准备的。对于其它 USART 则没有相应引脚，相关控制位及状态对 USART 无影响。

电源管理

USART 时钟不连续。在使用 USART 前，必须先使能电源管理控制器 (PMC) 中的 USART 时钟。但若应用中不需要 USART 时，USART 时钟可停止，并在需要时重新运行。该情况下，USART 恢复到停止前的状态。

对 USART 配置不需要使能 USART 时钟。



中断

USART中断线与高级中断控制器的一个内部中断源连接。使用USART中断请求前要先对AIC编程。注意，边沿敏感模式下不推荐使用 USART 中断线。

功能说明

USART 可管理多类型串行同步或异步通信。

它支持下列通信模式：

- 5 到 9 位全双工异步串行通信：
  - 高位或低位在先
  - 1、 1.5 或 2 位停止位
  - 奇检验、偶检验、标志、间隔或无
  - 接收器频率 8 或 16 倍重采样
  - 可选硬件握手
  - 可选调试解调器信号管理
  - 可选间断管理
  - 可选多点串行通信
- 高速 5 到 9 位全双工串行通信
  - 高位或低位在先
  - 1 或 2 位停止位
  - 奇检验、偶检验、标志、间隔或无
  - 接收器频率 8 或 16 倍重采样
  - 可选硬件握手
  - 可选调试解调器信号管理
  - 可选间断管理
  - 可选多点串行通信
- 含驱动器控制信号的 RS485
- ISO7816，与智能卡连接的 T0 或 T1 协议
  - NACK 处理，有复制与反复限制的错误计数器
- 红外 IrDA 调制解调
- 测试模式
  - 远程回环、本地回环、自动回应

波特率发生器

波特率发生器给接收器与发送器提供名为波特率时钟的位周期时钟。

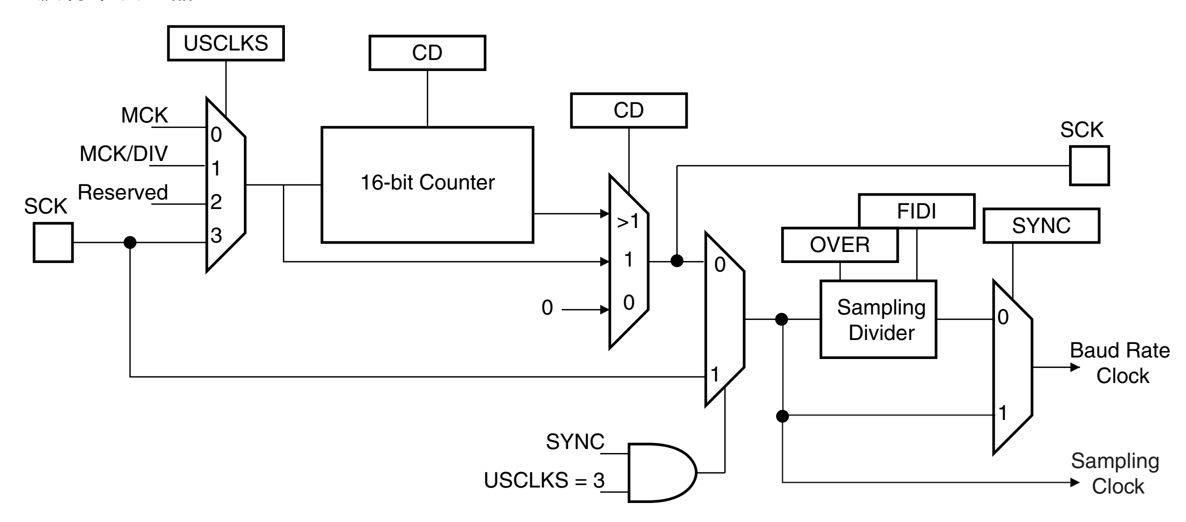
波特率发生器时钟源可设置模式寄存器 (US_MR) 的 USCLKS 域为下列状态：

- 主机时钟 MCK
- 主机时钟分频，分频因子由产品确定，通常情况下为 8
- 外部时钟，在 SCK 引脚有效

波特率发生器是基于波特率发生器寄存器 (US_BRGR) 中 CD 域编程的 16 位分频器。若 CD 为 0，波特率发生器不产生时钟；若 CD 为 1，分频器被绕过，并失效。

若选择外部 SCK 时钟，SCK 引脚高低电平持续时间必须比一个主机时钟 (MCK) 周期长。MCK 频率至少是 SCK 上信号频率的 4.5 倍。

Figure 174. 波特率发生器



异步模式下的波特率

若 USART 工作在异步模式下，选定的时钟首先除以 US_BRGR 寄存器中 CD 域的值。所得到的时钟作为接收器的采样时钟，再根据 US_MR 寄存器中的 OVER 位决定被 16 或 8 分频。

若 OVER 为 1，接收器采样时钟为波特率时钟的 8 倍；若 OVER 清零，接收器采样时钟为波特率时钟的 16 倍。

下列公式用来计算波特率：

$$Baudrate = \frac{SelectedClock}{(8(2 - Over)CD)}$$

假设 MCK 工作再最高时钟频率下，且 OVER 为 1，上式给出了 MCK 8 分频后的最大波特率。

波特率计算示例

Table 74 给出在不同时钟源频率下得到波特率为 38400 的 CD 计算方法。表中同时也给出了实际结果及误差。

Table 74. 波特率示例 (OVER = 0)

源时钟	期望波特率	计算结果	CD	实际波特率	误差
MHz	Bit/s			Bit/s	
3 686 400	38 400	6.00	6	38 400.00	0.00%
4 915 200	38 400	8.00	8	38 400.00	0.00%
5 000 000	38 400	8.14	8	39 062.50	1.70%
7 372 800	38 400	12.00	12	38 400.00	0.00%
8 000 000	38 400	13.02	13	38 461.54	0.16%
12 000 000	38 400	19.53	20	37 500.00	2.40%
12 288 000	38 400	20.00	20	38 400.00	0.00%
14 318 180	38 400	23.30	23	38 908.10	1.31%
14 745 600	38 400	24.00	24	38 400.00	0.00%
18 432 000	38 400	30.00	30	38 400.00	0.00%
24 000 000	38 400	39.06	39	38 461.54	0.16%
24 576 000	38 400	40.00	40	38 400.00	0.00%
25 000 000	38 400	40.69	40	38 109.76	0.76%
32 000 000	38 400	52.08	52	38 461.54	0.16%
32 768 000	38 400	53.33	53	38 641.51	0.63%
33 000 000	38 400	53.71	54	38 194.44	0.54%
40 000 000	38 400	65.10	65	38 461.54	0.16%
50 000 000	38 400	81.38	81	38 580.25	0.47%
60 000 000	38 400	97.66	98	38 265.31	0.35%
70 000 000	38 400	113.93	114	38 377.19	0.06%

波特率由下式计算得到：

$$BaudRate = MCK / CD \times 16$$

波特率误差由下式计算得到。建议误差应低于 5%。

$$Error = 1 - \left(\frac{ExpectedBaudRate}{ActualBaudRate}\right)$$

同步模式下的波特率

若 USART 工作在同步模式下，选定的时钟除以 US_BRGR 寄存器中 CD 域的值。

$$BaudRate = \frac{SelectedClock}{CD}$$

同步模式下，若选择外部时钟 (USCLKS = 3)，时钟由 USART SCK 引脚信号提供。不需分频。写入 US_BRGR 中的值无效。系统时钟频率至少是外部时钟频率的 4.5 倍。

当同时选择外部时钟 SCK 与内部分频时钟 (MCK/DIV) 时，若用户要 SCK 引脚信号占空比为 50:50，则 CD 值为偶数。若选择内部时钟 MCK，即使 CD 值为奇数， SCK 引脚信号占空比为 50:50。

ISO 7816 模式下波特率

ISO7816 用下式定义比特率：

$$B = \frac{Di}{Fi} \times f$$

其中：

- B 为比特率
- Di 为比特率调整因子
- Fi 为时钟频率分频因子
- f 为 ISO7816 时钟频率 (Hz)

Di 是一个 4 位二进制值，称为 DI，见 Table 75。

Table 75. D 的二进制与十进制值

DI 域	0001	0010	0011	0100	0101	0110	1000	1001
Di (十进制)	1	2	4	8	16	32	12	20

Fi 是一个 4 位二进制值，称为 FI，见 Table 76。

Table 76. Fi 的二进制与十进制值

FI 域	0000	0001	0010	0011	0100	0101	0110	1001	1010	1011	1100	1101
Fi (十进制	372	372	558	744	1116	1488	1860	512	768	1024	1536	2048

Table 77 给出 Fi/Di 比值，在 ISO7816 时钟与波特率时钟间。

Table 77. 可能的 Fi/Di 比值

Fi/Di	372	558	774	1116	1488	1806	512	768	1024	1536	2048
1	372	558	744	1116	1488	1860	512	768	1024	1536	2048
2	186	279	372	558	744	930	256	384	512	768	1024
4	93	139.5	186	279	372	465	128	192	256	384	512
8	46.5	69.75	93	139.5	186	232.5	64	96	128	192	256
16	23.25	34.87	46.5	69.75	93	116.2	32	48	64	96	128
32	11.62	17.43	23.25	34.87	46.5	58.13	16	24	32	48	64
12	31	46.5	62	93	124	155	42.66	64	85.33	128	170.6
20	18.6	27.9	37.2	55.8	74.4	93	25.6	38.4	51.2	76.8	102.4

若 USART 配置为 ISO7816 模式，模式寄存器 (US_MR) 中的 USCLKS 域指定的时钟首先除以波特率发生器寄存器 (US_BRGR) 中的 CD 编程值。得到的时钟提供给 SCK 引脚，作为智能卡时钟输入。即 CLKO 位可在 US_MR 中设置。

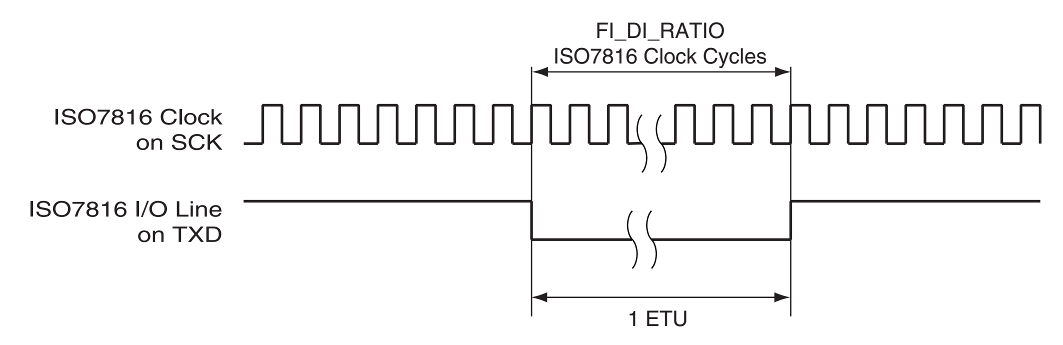
该时钟由 FI_DI_Ratio 寄存器 (US_FIDI) 中的 FI_DI_RATIO 域值分频。由 ISO7816 模式下可除高达 2047 的采样除法器执行。Fi/Di 比率必须为整数，且用户必须尽量将 FI_DI_RATIO 值接近于期望值。

FI_DI_RATIO 域复位值为 0x174 (十进制 372) ，这是 ISO7816 时钟与比特率间公共的除数 (Fi = 372， Di = 1)。



Figure 175 给出了与位时钟与 ISO 7816 时钟相关的关系。

Figure 175. 基本时间单元 (ETU)



接收器与发送器控制

复位后接收器禁用。用户必须通过设置控制寄存器 (US_CR) 的 RXEN 位使能接收器。但接收器寄存器在接收器时钟使能前可编程。

复位后发送器禁用。用户必须通过设置控制寄存器 (US_CR) 的 TXEN 位使能发送器。但发送器寄存器在发送器时钟使能前可编程。

发送器与接收器可一起或分别使能。

任意时刻，软件可通过分别置位 US_CR 寄存器中的 RSTRX 与 RSTTX 执行 USART 接收器或发送器复位。软件复位与硬件复位效果相同。复位时，不管是接收器还是发送器，通信立即停止。

用户也可通过设置 US_CR 寄存器中的 RXDIS 与 TXDIS 独立禁用接收器与发送器。若在接收字符时接收器禁用，USART 等待当前字符接收结束，再停止接收。若发送器工作时禁用，USART 等待当前字符及存于发送编程寄存器 (US_THR) 中的字符发送完成 。若编程设置了时间保障，它将正常处理。

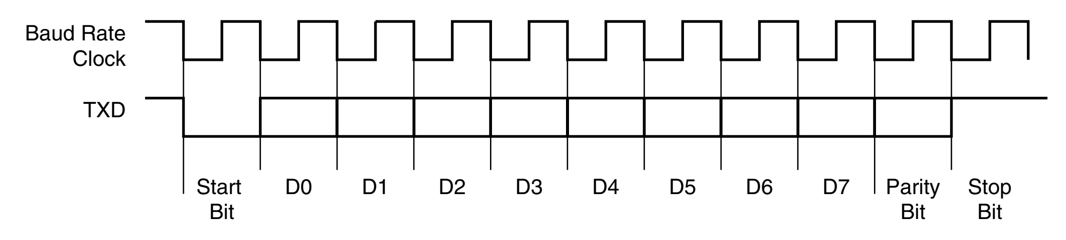
同步与异步模式

发送器操作

在同步与异步模式下 (SYNC = 0 或 SYNC = 1)，发送器操作相同。 1 个起始位，最多 9 个数据位， 1 个可选的奇偶校验位及最多 2 个停止位，在每个串行设置下降沿由 TXD 引脚移出。

数据位数目由 US_MR 寄存器的 CHRL 域及 MODE9 位决定。设置 MODE 9 位时，不管 CHRL 域设置，数据位为 9 位。奇偶校验位由 PAR 域设置。可配置为奇检验、偶检验、空检验、标志检验或无校验位。MSBF 域配置首先发送的位。若写入 1，将先发送最高位；若写入 0，将先发送最低位。停止位数目由 NBSTOP 域选择。异步模式下支持 1.5 停止位。

Figure 176. 字符发送

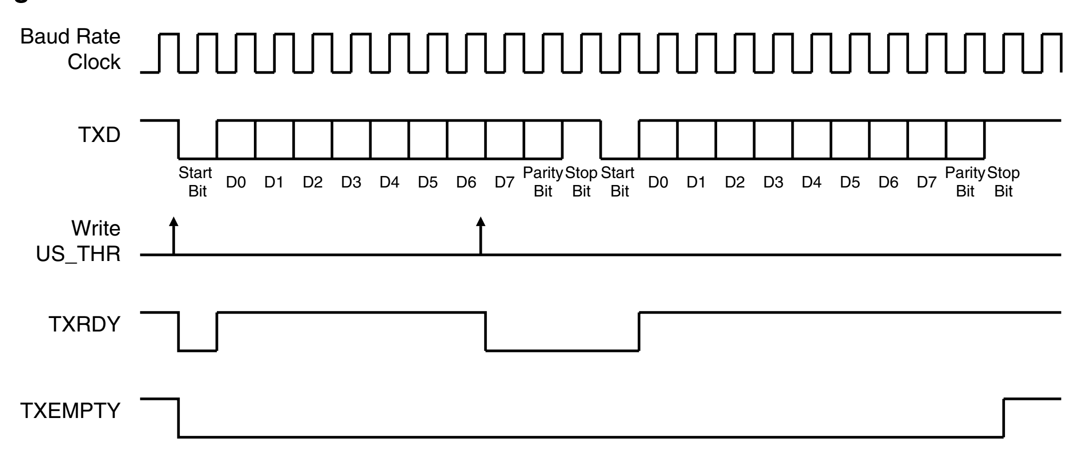


字符发送到发送保持寄存器 (US_THR) 中。发送器在通道状态寄存器 (US_CSR) 中有 2 个状态位：TXRDY (发送就绪) 表示 US_THR 空且 TXEMPTY，即所有写入 US_THR 中的字符已开

始处理。当当前字符处理完成，最后写入US_THR的字符送入发送器移位寄存器中，且US_THR变空，因此 TXRDY 升高。

发送器禁用后， TXRDY 与 TXEMPTY 位均为低。当 TXRDY 有效，在 US_THR 中写入字符无效，且写入数据丢失。

Figure 177. 发送器状态



异步接收器

若 USART 工作在异步模式下 (SYNC = 0)，接收器对 RXD 输入线重采样。重采样为 16 或 8 倍波特率时钟，由 US_MR 中的 OVER 位设置。

接收器对 RXD 线采样。若在 1.5 个比特时间内采样值为 0，即表示检测到起始位，然后数据、校验位、停止位等以比特速率时钟采样。

若重采样为 16(OVER 为 0)， 8 次采样结果均为 0，表示检测到起始位。然后，每 16 个采样时钟周期中对数据、校验位、停止位依次采样。若重采样为 8 (OVER 为 1)， 4 次采样结果均为 0，表示检测到起始位。然后，每 8 个采样时钟周期中对数据、校验位、停止位依次采样。

数据位数、最先发送位及校验模式选择的域及位与发送器相同，即分别为 CHRL、 MODE9、MSBF及PAR。停止位数对接收器无效，因为无论NBSTOP域为何值，接收器都只确认1个停止位，因此会出现发送器与接收器间的重同步。此外，接收器在检测到停止位后即开始寻找新的起始位，因此当发送器只有 1 个停止位时也能实现重同步。

Figure 178 与 Figure 179 给出当 USART 工作在异步模式下的起始检测及字符接收。



Figure 178. 异步起始检测

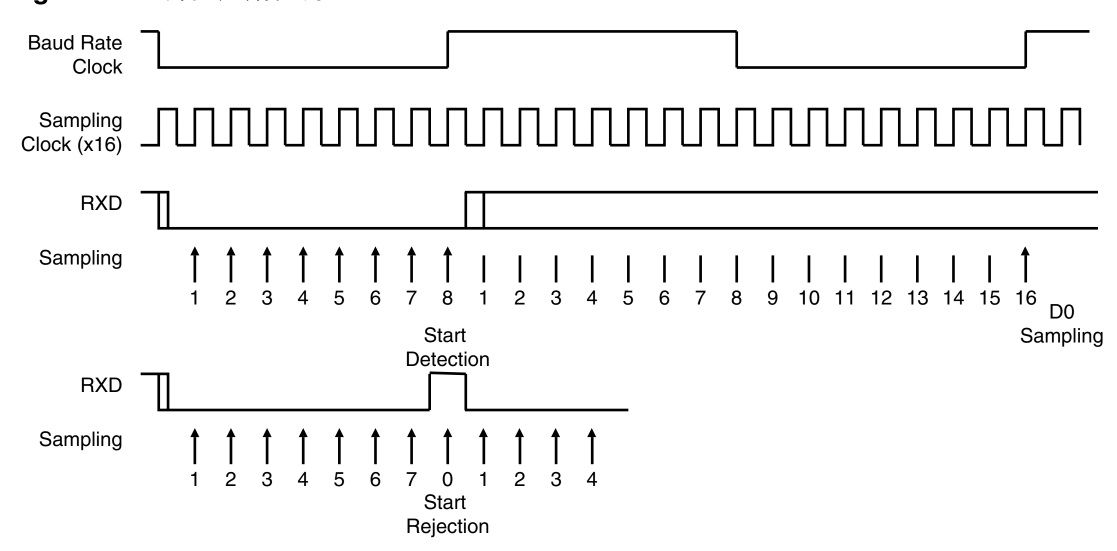
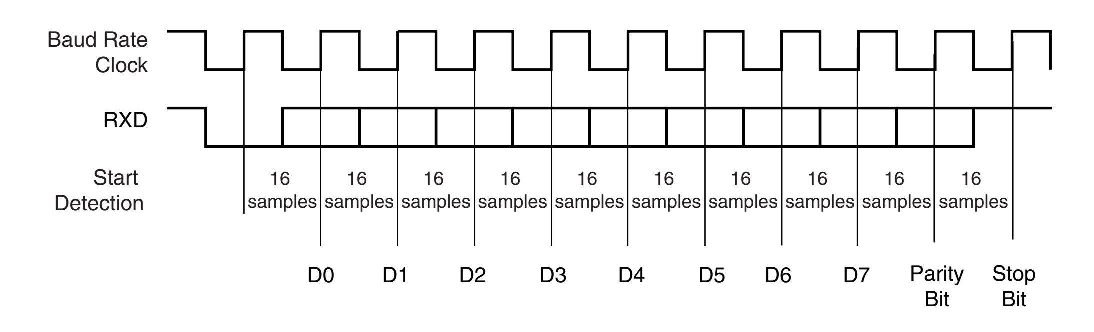


Figure 179. 异步字符接收



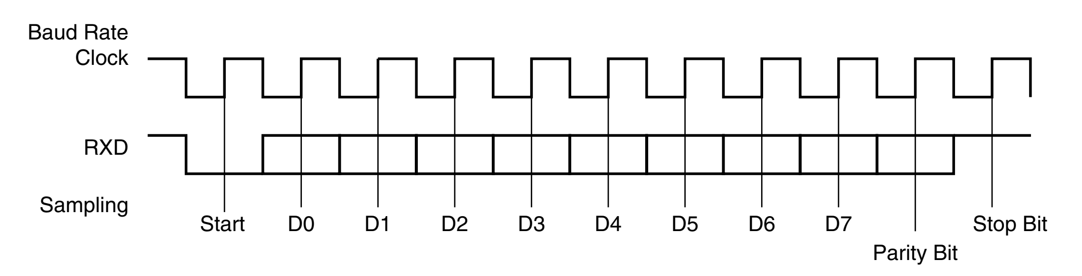
同步接收器

同步模式下 (SYNC = 1)，接收器在波特率时钟的每个上升沿对 RXD 信号采样。若检测到低电平，则为起始位。所有数据位、校验位及停止位依次采样，接收器等待下一个起始位。同步模式提供高速传输能力。

域及位的配置与异步模式下相同。

Figure 180 给出同步模式下的字符接收。

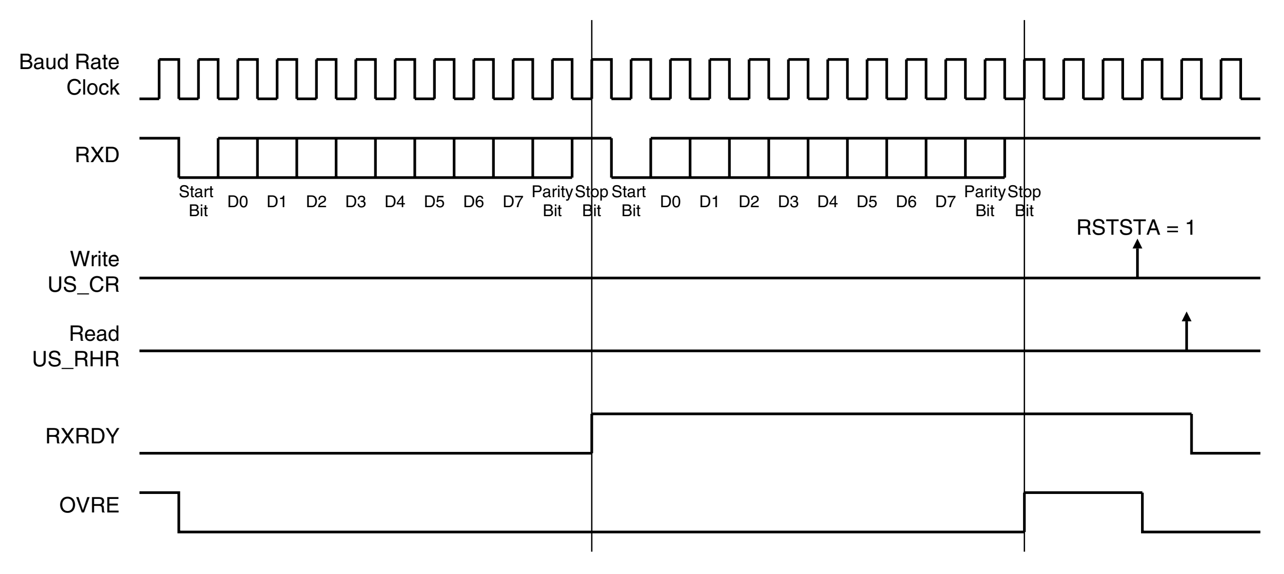
Figure 180. 同步模式字符接收



接收器操作

当字符接收完成，它将传输到接收保持寄存器 (US_RHR)，且状态寄存器 (US_CSR) 的 RXRDY 位变高。若字符完成而 RXRDY 置位，OVRE (溢出错误) 位置位。最后的字符传输到 US_RHR 并覆盖上一个字符。对控制寄存器 (US_CR) 的 RSTSTA (复位状态) 位写 1 将清除 OVRE 位。

Figure 181. 接收器状态



奇偶检验

USART通过对模式寄存器(US_MR)PAR域的设置，可支持5种奇偶检验模式。PAR域还使能多点模式，这将在其它章节予以讨论 。产生奇偶检验位并支持错误检测。

若选择偶检验，当发送器发送 1 的数目为偶数时，校验位发生器产生校验位为 1 ；当发送器发送 1 的数目为奇数时，校验位发生器产生校验位为 0。相应的，接收器校验位检测器对收到的 1 计数，若与采样到的校验位不符，则报告奇偶检验错误。若选择奇检验，当发送器发送 1 的数目为偶数时，校验位发生器产生校验位为 0 ；当发送器发送 1 的数目为奇数时，校验位发生器产生校验位为 1。相应的，接收器校验位检测器对收到的 1 计数，若与采样到的校验位不符，则报告奇偶检验错误。若使用标志奇偶检验，对于所有字符，奇偶检验发生器都将奇偶校验位置 1。若接收器采样得到的校验位为 0，接收器校验位检测器报告检验错误。若使用空奇偶检验，对于所有字符，奇偶检验发生器都将奇偶校验位置 0。若接收器采样得到的校验位为 1，接收器校验位检测器报告检验错误。若奇偶检验禁用，发送器不产生校验位，接收器也不报告奇偶检验错误。

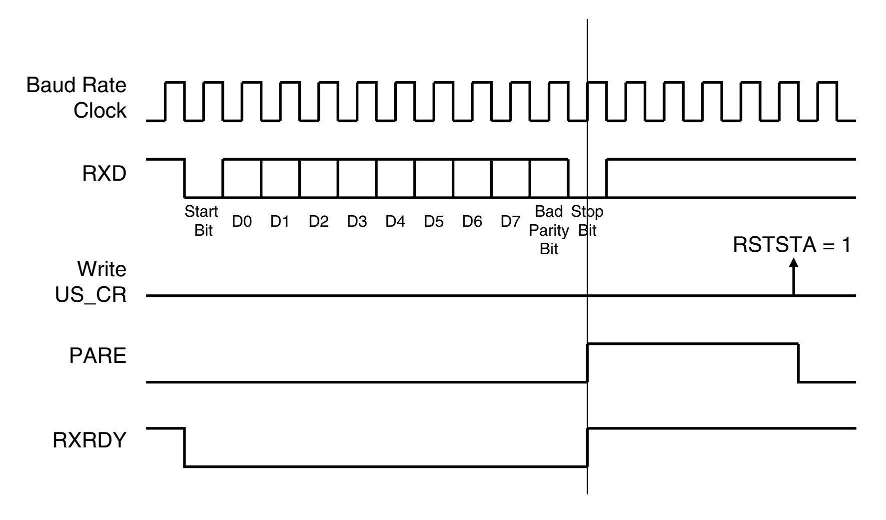
Table 78给出根据USART配置，字符0x41 (ASCII字符 “A”)的奇偶校验位的例子。由于有两位为 1，当奇偶检验为奇数时加 1 位，为偶数时加 0 位。

Table 78. 奇偶校验位示例

字符	十六进制	二进制	校验位	检验模式
A	0x41	0100 0001	1	奇检验
A	0x41	0100 0001	0	偶检验
A	0x41	0100 0001	1	标志检验
A	0x41	0100 0001	0	空检验
A	0x41	0100 0001	无	无

当接收器检测到奇偶检验误差，它设置通道状态寄存器 (US_CSR) 的 PARE (奇偶检验错误) 位。对控制寄存器 (US_CR) 的 RSTSTA 位写 1 将清除 PARE 位。 Figure 182 给出奇偶校验位的置位与清零。

Figure 182. 奇偶检验错误



多点模式

若模式寄存器 (US_MR) 的 PAR 域编程值为 0x6 或 0x7， USART 运行在多点模式下。该模式区分数据字符与地址字符。当奇偶校验位为 0 时发送数据；当奇偶校验位为 1 时发送地址。

若 USART 配置为多点模式，当奇偶校验位为高时，接收器将 PARE 位置位，当控制寄存器 SENDA 位为 1 时，若奇偶校验位为高，发送器可发送字符。

为处理奇偶检验错误，当控制寄存器 RSTSTA 位写 1 时， PARE 位清零。

当SENDA写入US_CR时，发送器发出地址字节(奇偶校验位置位)。此时，下一个写入US_THR 的字节将作为地址来发送。没有写 SENDA 命令的任何写入 US_THR 的字符将正常被发送 (奇偶校验位为 0) 。

发送器时间保障

时间保障特性使能 USART 与慢速远程器件的连接。

时间保障功能使能发送器 TXD 线上在两字符间插入空闲状态。该空闲状态实际上是一个长停止位。

空闲状态的持续时间由发送时间保障寄存器 (US_TTGR) 的 TG 域编程设定。当该域编程值为零，则不产生时间保障。否则，在每次发送了停止位及 TG 中指定的位数周期后，在 TXD 线上发送器保持高电平。

如 Figure 183 所示， TXRDY 与 TXEMPTY 状态位的行为可由时间保障改变。 TXRDY 只有在下一字符起始位发送后才变高。若字符已写入 US_THR，则时间保障传输期间 TXRDY 保持为 0。由于时间保障是当前传输的一部分，因此 TXEMPTY 为低要保持到时间保障传输结束。

Figure 183. 时间保障操作

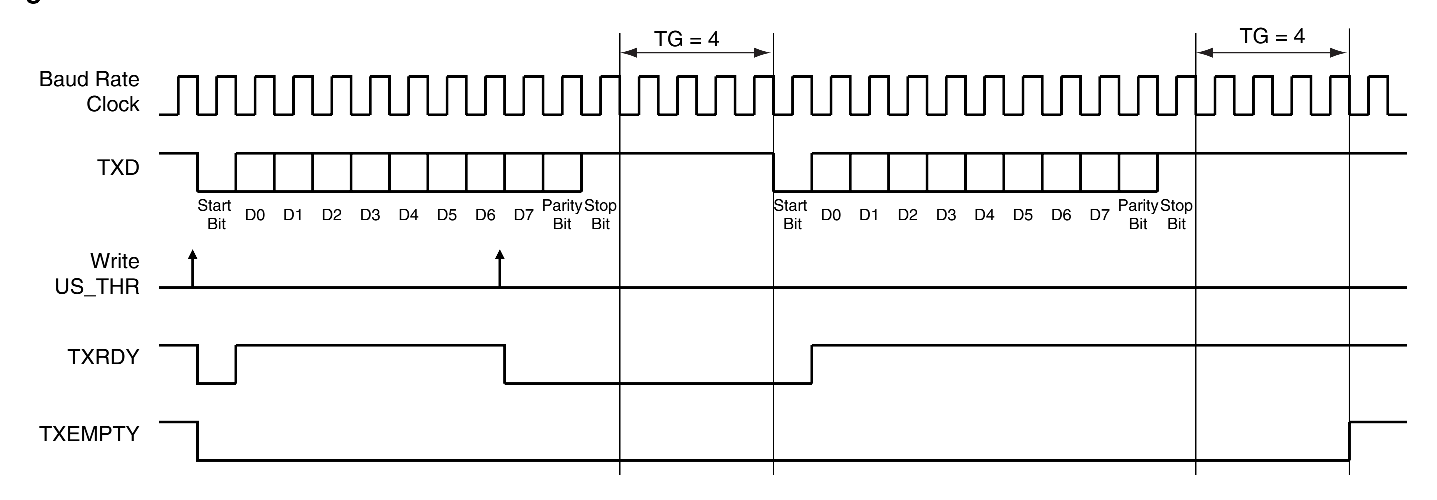


Table 79 列出对应不同波特率，发送器能处理的时间保障周期的最大值。

Table 79. 不同波特率下时间保障最大值

波特率	位时间	时间保障
bit/sec	µs	ms
1 200	833	212.50
9 600	104	26.56
14400	69.4	17.71
19200	52.1	13.28
28800	34.7	8.85
33400	29.9	7.63
56000	17.9	4.55
57600	17.4	4.43
115200	8.7	2.21

接收器超时

接收器超时支持对长度可变的帧处理。该通信检测 RXD 线上的空闲状态。当检测到超时，通道状态寄存器 (US_CSR) 的 TIMEOUT 位变高并产生中断，以告知驱动器帧结束。

超时延迟周期 (接收器等待新字符时间) 在接收器超时寄存器 (US_RTOR) 的 TO 域编程。若 TO 域编程为 0，接收器超时禁用，将检测不到超时。 US_CSR 寄存器中 TIMEOUT 位保持为 0。否则，接收器将 TO 中值载入一个 16 位计数器。该计数器在每比特周期中自减并在收到新字符后重载。若计数器达到 0，状态寄存器中 TIMEOUT 位变高。

用户可：

- 至少收到一个字符后，当检测到超时时中断。通过在控制寄存器 (US_CR) 的 STTTO (启动超时) 位写 1 实现。
- 没有收到字符时周期中断。通过在 US_CR 中 RETTO (重载与启动超时) 位写 1 实现。

若 STTTO 执行，计数器时钟在收到第一个字符前停止。帧启动前 RXD 的空闲状态不提供超时。这样可防止周期性中断并在检测到 RXD 为空闲状态时使能帧结束等待。

若 RETTO 执行，计数器开始从 TO 值向下计数。因而产生周期性中断使可处理用户超时 ，例如当键盘上没有键入时。



Figure 184 给出接收器超时特性框图。

Figure 184. 计数器超时框图

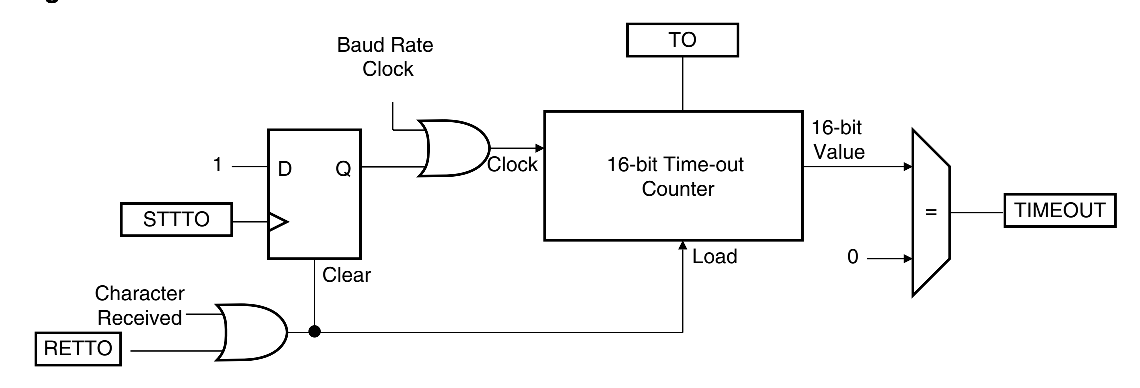


Table 80 给出某些标准波特率下的最大超时周期。

Table 80. 最大超时周期

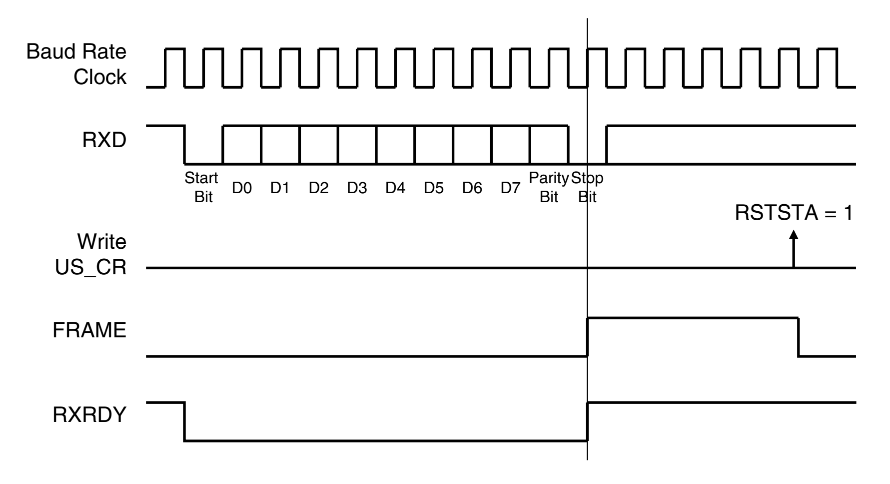
波特率	位时间	超时
bit/sec	µs	ms
600	1 667	109 225
1 200	833	54 613
2 400	417	27 306
4 800	208	13 653
9 600	104	6 827
14400	69	4 551
19200	52	3 413
28800	35	2 276
33400	30	1 962
56000	18	1 170
57600	17	1 138
200000	5	328

帧错误

接收器可检测帧错误。当检测到收到字符的停止位为 0 时产生帧错误。当接收器与发送器为完全同步时可能出现。

帧错误由 US_CSR 寄存器的 FRAME 位表示。检测到帧错误时， FRAME 位在停止位中间出现。通过将 US_CR 寄存器的 RSTSTA 位写为 1 将其清除。

Figure 185. 帧错误状态



发送间断

用户可请求发送器在 TXD 线上产生间断条件。使得至少在一个完整字符时间中 TXD 线为低。这与校验位及停止位均为 0 的 0x00 字符的发送情况相同。无论如何，发送器至少保证 TXD 线在一个完整的字符传输时间内为低，直到用户请求将间断条件删除。

将 US_CR 寄存器 STTBRK 位写 1 将发送一个间断。这可在任何时间执行，即使发送器为空 (移位寄存器及 US_THR 中均无字符) 或字符正在发送。若字符移出时出现间断请求 , 在 TXD 线变低前字符完成。

一旦需要 STTBRK 命令，在间断完成前忽略 STTBRK 命令。

通过在 US_CR 寄存器的 STPBRK 位写 1 将删除间断条件。若在最小间断持续时间 (一字符，包括起始位、数据位、校验位及停止位) 结束前请求 STPBRK，发送器保证间断条件完成。

发送器将间断视为一个字符 , 即只有当 US_CSR 寄存器中 TXRDY 为 1 时 ,STTBRK 与 STPBRK 命令才被考虑，如处理一个字符一样，间断条件启动将清除 TXRDY 与 TXEMPTY 位。

将 US_CR 寄存器中 STTBRK 与 STPBRK 位写为 1 结果无法预测。所有前面没有 STTBRK 命令的 STPBRK 命令请求将被忽略。当间断挂起时写入发送保持寄存器但未启动的字节将忽略。

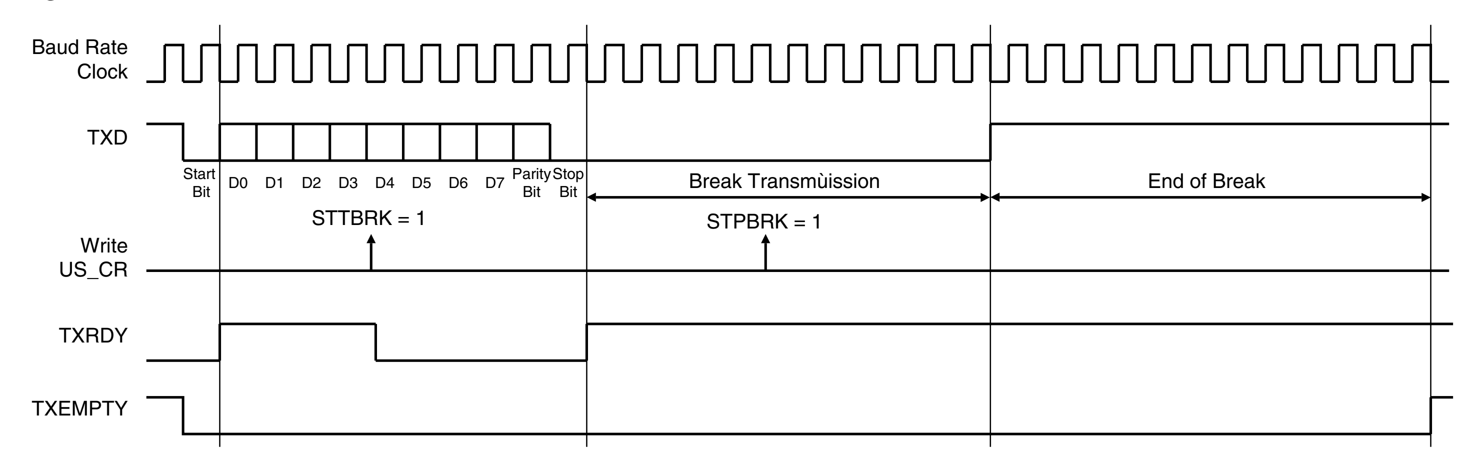
间断条件后，最少 12 比特时间内，发送器将 TXD 线变回 1。因此，发送器保证远程接收器正确检测到间断结束并开始下一个字符。若时间保障值大于 12，TXD 线将在时间保障周期内保持高电平。

将 TXD 线保持这一周期后，发送器恢复正常工作。

Figure 186 给出 TXD 线上开始间断 (STTBRK) 与停止间断 (STP BRK) 命令的作用。



Figure 186. 间断传输



接收间断

当所有数据、检验及停止位均为低时，接收器检测到间断条件。这与当数据为 0x00 但 FRAME 为低时的检测结果相当。

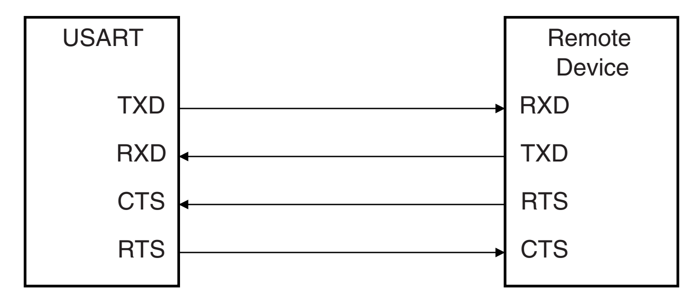
当检测到低电平的停止位时，接收器将US_CSR寄存器的RXBRK位使能，该位可通过在US_CR 寄存器的 RSTSTA 位写 1 来清零。

异步工作模式下检测到至少 2/16 的比特周期为高电平，或在同步工作模式下有一个采样值为高，表明接收间断结束。间断结束也可通过 RXBRK 置位来实现。

硬件握手

USART 有硬件握手段外流控制特性。RTS 与 CTS 引脚与远程器件连接，如 Figure 187 所示。

Figure 187. 与远程器件连接的硬件握手



在 US_MR 寄存器 USART_MODE 域写入 0x2，则 USART 将执行硬件握手操作。

除接收器对 RTS 引脚及发送器对 CTS 引脚电平改变外，硬件握手使能后的 USART 操作与标准同步或异步模式下相同 。使用该模式需要用 PDC 通道接收数据。发送器在任何情况下均可处理硬件握手。

Figure 188给出当硬件握手使能时接收器的操作。若接收器禁用且来自PDC通道的RXBUFF (接收缓冲满）状态为高， RTS 引脚拉高。通常当 CTS 引脚（由 RTS 驱动）为高时远程器件不会开始发送。一旦接收器使能， RTS 变低，则表示远程器件可启动发送。给 PDC 定义一个新缓冲器，将 RXBUFF 状态位清零，则 RTS 引脚随之变低。

Figure 188. 硬件握手时接收器工作情况

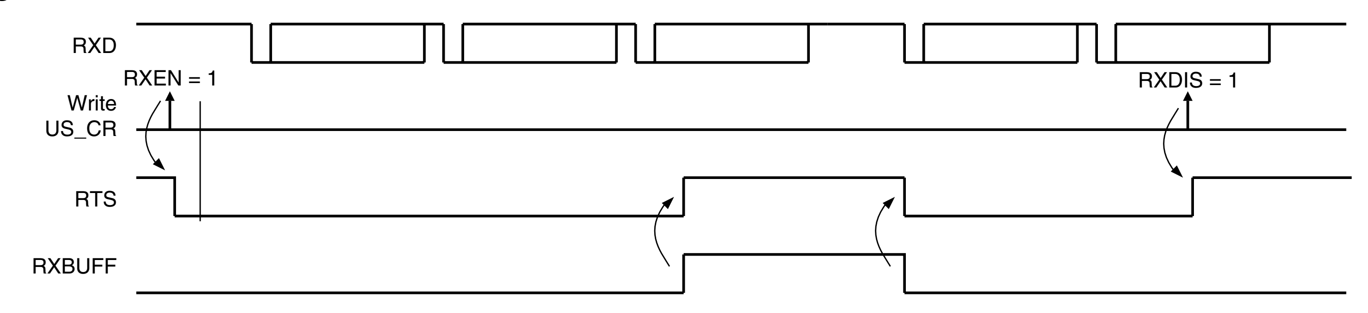
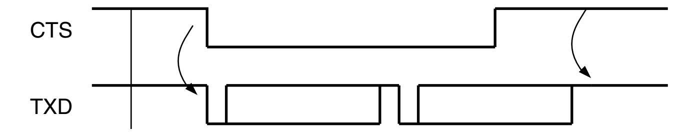


Figure 189 给出硬件握手使能后 , 发送器如何操作。CTS 引脚禁用发送器。若有字符正在处理， 则在当前字符处理完成后将发送器禁用，一旦 CTS 变低即开始下一个字符的发送。

Figure 189. 硬件握手时发送器工作情况



ISO7816 模式

USART有一个与ISO7816兼容模式。该模式允许与智能卡连接并可通过ISO7816链接与安全访问模块 (SAM) 通信。支持 ISO7816 规范定义的 T = 0 与 T = 1 协议。

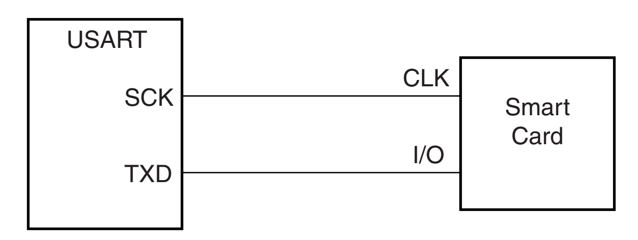
将 US_MR 寄存器 USART_MODE 域写 0x4，将设置 USART 在 ISO7816 的 T = 0 模式下工作；若 USART_MODE 域写 0x5，则 USART 在 ISO7816 的 T = 1 模式下工作。

ISO7816 模式概述

ISO7816为一条双向线的半双工通信。波特率由远程器件提供的时钟分频提供(见"波特率发生器" on page 396)。

USART与智能卡的连接见 Figure 190。TXD线变为双向，波特率发生器由SCK引脚向ISO7816提供时钟。由于 TXD 引脚变为双向，其输出由发送器输出驱动，但只有当发送器激活时其输入为接收器输入。由于 USART 产生时钟，因此它被视为通信主机。

Figure 190. 智能卡与 USART 的连接



无论工作在 ISO7816 的 T = 0 或 T = 1 模式下，字符格式固定。不管 CHRL、 MODE9、 PAR 及 CHMODE 域中是什么值，其配置为 8 位数据位，偶检验及 1 或 2 位停止位。 MSBF 可设置发送是高位在先还是低位在先。

由于通信不是双向的，USART 不能同时在发送器与接收器模式下操作。因而必须根据需要使能或禁用接收器或发送器。 ISO7816 模式下同时使能发送器与接收器结果无法预知。

ISO7816 规范定义了一个反转发送格式。字符的数据位必须在 I/O 线上以其负值发送。USART 不支持该格式，因此在将其写入发送保持寄存器 (US_THR) 前或由接收保持寄存器 (US_RHR) 读出后必须进行一个额外的或操作。

T = 0 协议

T = 0 协议中，字符由一个起始位、八个数据位、一个奇偶校验位及一个两比特时间的保障时间组成。在保障时间中，发送器移出比特但不驱动 I/O 线。

若无检验错误检测，保障时间内I/O线保持为 1，且发送器可继续发送下个字符，见 Figure 191。

若接收器检测到检验错误，在保障时间内它将 I/O 线驱动为 0，如 Figure 192 所示。该错误位也称为 NACK，即无应答。此时，由于保障时间长度未变而增加了一个 1 比特时间的错误位时间，因此字符加长 1 比特时间。

当 USART 为接收器且检测到错误，它不会将字符载入接收保持寄存器 (US_RHR)。而是适当设置状态寄存器 (US_SR) 的 PARE 位以使软件处理该错误。

Figure 191. 没有检验错误的 T = 0 协议

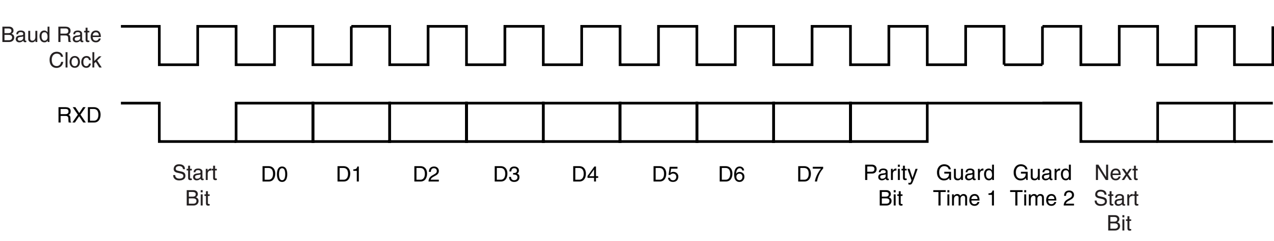
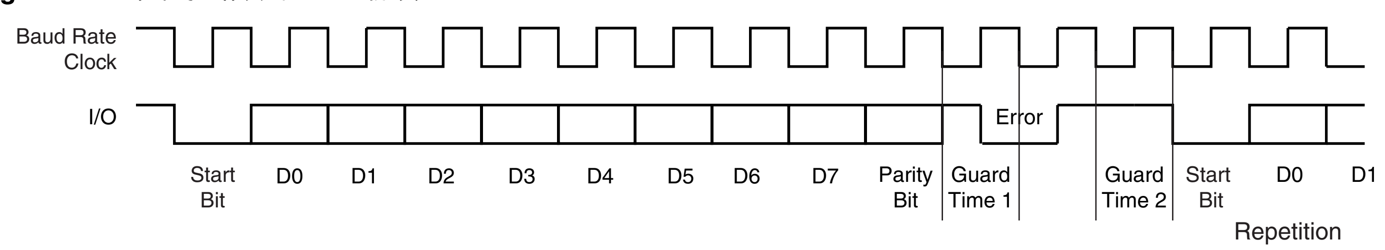


Figure 192. 有检验错误的 T = 0 协议



接收错误计数器

USART 接收器还记录错误总数。可从错误数目寄存器 (US_NER) 中读出。NB_ERRORS 域可记录 255 个错误。读取 US_NER 自动清除 NB_ERRORS 域。

接收 NACK 抑制

USART 可配置为抑制错误。通过置位 US_MR 的 INACK 位实现。若 INACK 为 1 ,即使检测到检验位 ,I/O 线上没有错误信号 ,但 US_SR 寄存器中 INACK 位置位。可通过写控制寄存器 (US_CR) 的 RSTNACK 位为 1 来清除 INACK 位。

此外，若 INACK 置位，接收到的错误字符保存在接收保持寄存器中，就像没有错误出现。但 RXRDY 位不会升高。

发送字符复制

当 USART 正在发送字符并得到 NACK 时，在移到下一个字符前，它可自动复制该字符。通过在 US_MR 寄存器 MAX_ITERATION 域中写入大于 0 的值使能复制。每个字符最多可发送八次 ；第一次发送加七次复制发送。

若 MAX_ITERATION 不等于 0， USART 复制字符的次数与 MAX_ITERATION 中值相同。

当 USART 复制次数达到 MAX_ITERATION 值时，通道状态寄存器 (US_CSR) 中 ITERATION 位置位。若复制字符得到接收器应答，复制停止并将迭代计数器清零。

US_CSR 寄存器中 ITERATION 位可通过在控制寄存器的 RSIT 位写 1 清零。

禁用连续接收 NACK

接收器可限制返回远程发送器的连续 NACK。通过设置 US_MR 寄存器的 DSNACK 位实现。NACK发送的最多数目在MAX_ITERATION域中编程。一旦达到MAX_ITERATION ,认为字符正确，线上发送应答并将通道状态寄存器的 ITERATION 位置位。

T = 1 协议

当工作在 ISO7816 的 T = 1 协议下，发送操作与只有一位停止位的异步格式相似。当发送时产生奇偶校验位 ,在接收时对其检测。奇偶错误检测通过设置通道状态寄存器 (US_CSR) 的 PARE 位实现。

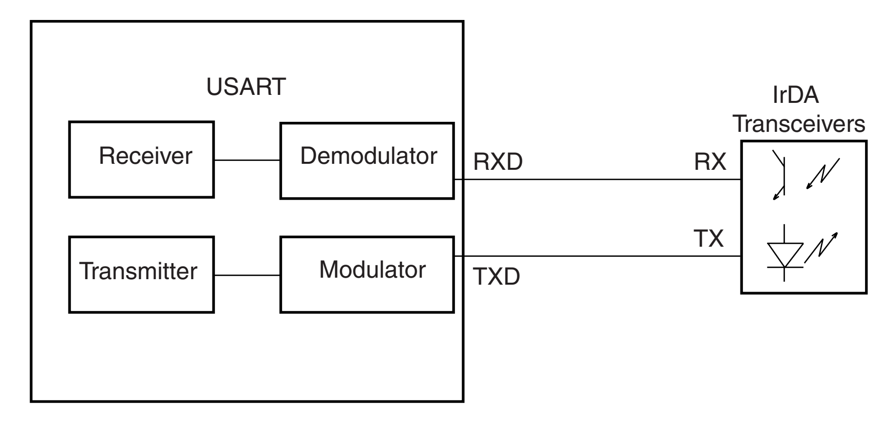
IrDA 模式

USART 的 IrDA 模式支持半双工点对点无线通信。它内置了与红外收发器无连接的调制器和解调器 ,见 Figure 193。调制器与解调器适用于 IrDA 规范版本 1.1 ,支持的数据传输速度范围从 2.4 Kb/s 到 115.2 Kb/s。

在 US_MR 寄存器的 USART_MODE 域写 0x8 将使能 USART IrDA 模式。 IrDA 滤波寄存器 (US_IF) 可配置解调滤波器。USART 发送器与接收器工作在正常异步模式下 ,所有参数可访问。注意，调制器与解调器均处于激活状态。



Figure 193. 与 IrDA 收发器连接



接收器与发送器必须根据传输方向使能或禁用。

IrDA 调制

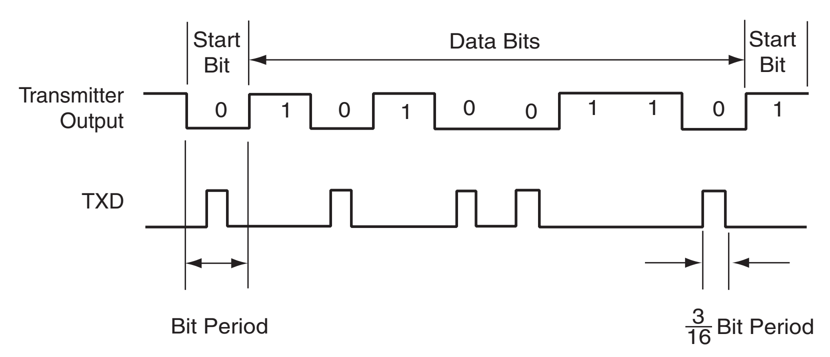
当波特率小于或等于 115.2 Kb/s，使用 RZI 调制方案。由一个 3/16 比特周期的轻脉冲表示 "0"。Table 81 中给出一些信号脉冲持续时间。

Table 81. IrDA 脉冲持续时间

波特率	脉冲持续时间 (3/16)
2.4 Kb/s	78.13 µs
9.6 Kb/s	19.53 µs
19.2 Kb/s	9.77 µs
38.4 Kb/s	4.88 µs
57.6 Kb/s	3.26 µs
115.2 Kb/s	1.63 µs

Figure 194 给出字符发送的示例。

Figure 194. IrDA 调制



IrDA 波特率

Table 82 给出一些 CD 值、波特率误差及脉冲持续时间的例子。注意，可接受的最高误差为 ±1.87%。

Table 82. IrDA 波特率误差

外设时钟	波特率	CD	波特率误差	脉冲时间
3 686 400	115 200	2	0.00%	1.63
20 000 000	115 200	11	1.38%	1.63
32 768 000	115 200	18	1.25%	1.63
40 000 000	115 200	22	1.38%	1.63
3 686 400	57 600	4	0.00%	3.26
20 000 000	57 600	22	1.38%	3.26
32 768 000	57 600	36	1.25%	3.26
40 000 000	57 600	43	0.93%	3.26
3 686 400	38 400	6	0.00%	4.88
20 000 000	38 400	33	1.38%	4.88
32 768 000	38 400	53	0.63%	4.88
40 000 000	38 400	65	0.16%	4.88
3 686 400	19 200	12	0.00%	9.77
20 000 000	19 200	65	0.16%	9.77
32 768 000	19 200	107	0.31%	9.77
40 000 000	19 200	130	0.16%	9.77
3 686 400	9 600	24	0.00%	19.53
20 000 000	9 600	130	0.16%	19.53
32 768 000	9 600	213	0.16%	19.53
40 000 000	9 600	260	0.16%	19.53
3 686 400	2 400	96	0.00%	78.13
20 000 000	2 400	521	0.03%	78.13
32 768 000	2 400	853	0.04%	78.13

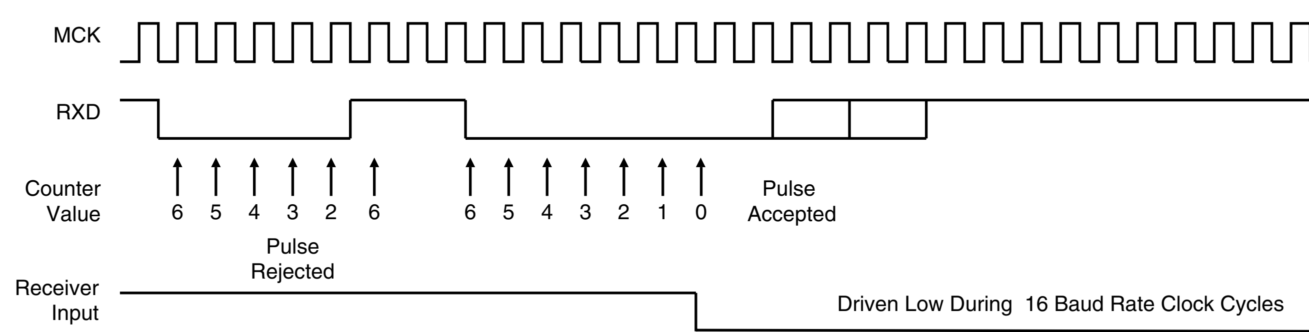
IrDA 解调器

解调器基于 IrDA 接收滤波器，包含一个由 US_IF 载入值的 8 位向下计数器。当检测到 RXD 引脚下降沿，滤波计数器以主机时钟 (MCK) 速度向下计数。若检测到 RXD 引脚上升沿，计数器停止并重新载入 US_IF 值。若计数器到达 0 仍未检测到上升沿，在一个比特时间内计数器输入拉低。

Figure 195 给出 IrDA 解调器的操作。



Figure 195. IrDA 解调器操作

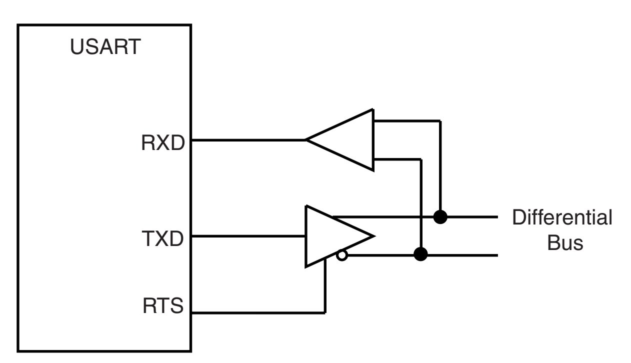


由于 IrDA 模式与 ISO7816 使用相同的逻辑，因此要注意 US_FIDI 中的 FI_DI_RATIO 域值必须大于 0，以确保 IrDA 通信操作正确。

RS485 模式

USART 的 RS485 模式使能线驱动控制。在 RS485 模式下 ,USART 与异步或同步模式下操作相同,并可对所有参数进行配置。不同处在于当发送器工作时将 RTS 引脚拉高。 RTS 引脚动作由 TXEMPTY 位控制。Figure 196 给出了 USART 与 RS485 总线的典型连接。

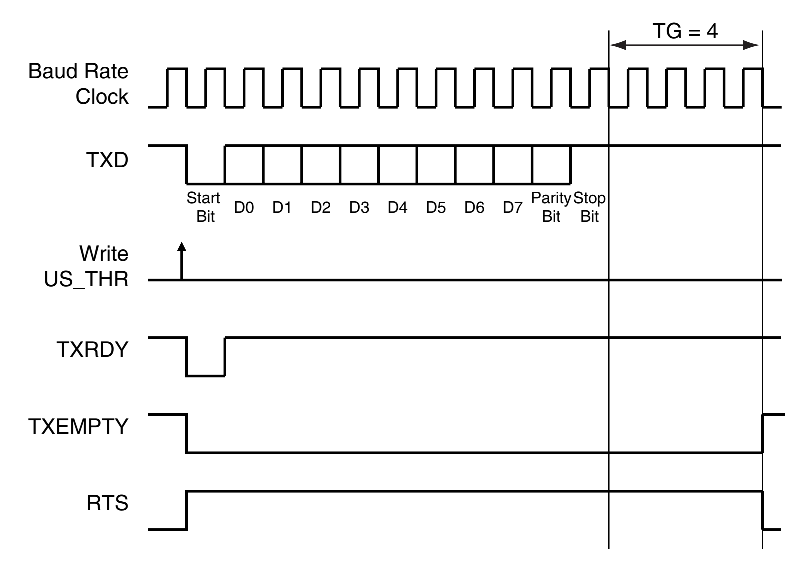
Figure 196. 与 RS485 总线的典型连接



在 US_MR 寄存器的 USART_MODE 域写入 0x1,可将 USART 设置为 RS485 模式。

RTS引脚电平与TXEMPTY位电平相反。注意 ,当时间保障编程时RTS引脚为高 ,因此在最后一个字符完成后线依然为高。 Figure 197 给出当时间保障使能,发送字符时的 RTS 波形。

Figure 197. 有时间保障的 RTS 驱动示例





调制解调模式

USART 调制解调模式使能下列信号控制:DTR (数据终端就绪)、DSR (时间设置就绪)、RTS (发送请求)、CTS (发送清零)、DCD (数据载波检测) 及 RI (环指示器)。在调制解调模式下，USART 作为 DTE (数据终端设备) 使用 ,驱动 DTR 与 RTS ,并可检测 DSR、DCD、CTS 及 RI 上电平变化。

在 US_MR 寄存器 USART_MODE 域写入 0x3，可将 USART 设置为调制解调模式。 USART 工作在调制解调模式模式下操作与异步模式下相同，可对所有参数进行配置。

Table 83 给出调制解调连接模式下对应的 USART 信号。

Table 83. 电路参考

USART 引脚	V24	CCITT	方向
TXD	2	103	由终端到调制解调器
RTS	4	105	由终端到调制解调器
DTR	20	108.2	由终端到调制解调器
RXD	3	104	由调制解调器到终端
CTS	5	106	由终端到调制解调器
DSR	6	107	由终端到调制解调器
DCD	8	109	由终端到调制解调器
RI	22	125	由终端到调制解调器

在控制寄存器 (US_CR) 的 RTSDIS、 RTSEN、 DTRDIS 与 DTREN 位分别置 1，可控制 RTS 与 DTR 输出引脚。禁用命令强制相应位为无效电平，即高电平；使能命令强制相应位为有效电平，即低电平。

对RI、DSR、DCD即CTS引脚进行电平检测。若检测到输入变化 ,将通道状态寄存器(US_CSR) 中对应的 RIIC、 DSRIC、 DCDIC 及 CTSIC 位置位并可触发中断。当读取 US_CSR 时，状态自动清除。此外，当检测到处于无效状态时， CTS 自动禁用发送器。若 CTS 变高时字符正在发送，则字符发送完成后，发送器才真正禁用。

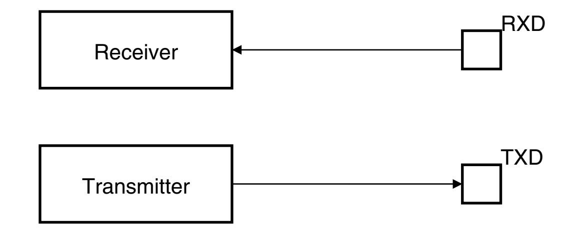
测试模式

USART 有三种不同的测试模式。内部回环可实现板上诊断。回环模式下， USART 接口引脚可不连接，并重新配置为内部或外部回环。

普通模式

普通模式下将 RXD 引脚与接收器输入连接，而发送器输出与 TXD 引脚连接。

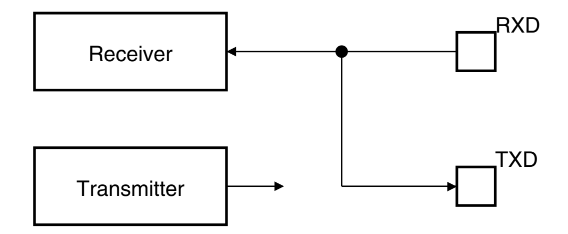
Figure 198. 普通模式配置



自动回应模式

自动回应模式允许位重发。当 RXD 引脚收到一位，它发送到 TXD 引脚，见 Figure 199。对发送器编程不影响 TXD 引脚。 RXD 引脚仍与接收器输入连接，因此接收器保持激活状态。

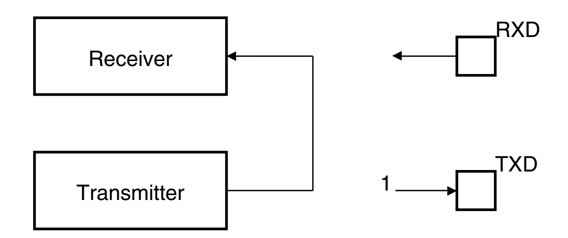
Figure 199. 自动回应



本地回环模式

本地回环模式下，发送器输出直接与接收器输入连接，如 Figure 200 所示。 TXD 与 RXD 引脚未使用。 RXD 引脚对接收器无效，而 TXD 引脚如空闲状态一样，始终为高。

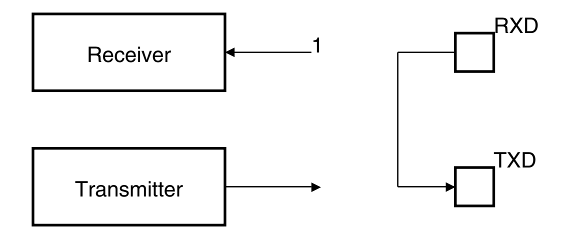
Figure 200. 本地回环



远程回环模式

远程回环模式下，直接将 RXD 引脚与 TXD 引脚连接，如 Figure 201 所示。发送器与接收器禁用无效。该模式允许位重发。

Figure 201. 远程回环





USART 用户接口

Table 84. USART 存储器映射

偏移	寄存器	名称	访问类型	复位状态
0x0000	控制寄存器	US_CR	只写	–
0x0004	模式寄存器	US_MR	读 / 写	–
0x0008	中断使能寄存器	US_IER	只写	–
0x000C	中断禁用寄存器	US_IDR	只写	–
0x0010	中断屏蔽寄存器	US_IMR	只读	0
0x0014	通道状态寄存器	US_CSR	只读	–
0x0018	接收器保持寄存器	US_RHR	只读	0
0x001C	发送器保持寄存器	US_THR	只写	–
0x0020	波特率发生器寄存器	US_BRGR	读 / 写	0
0x0024	接收器超时寄存器	US_RTOR	读 / 写	0
0x0028	发送器时间保障寄存器	US_TTGR	读 / 写	0
0x2C-0x3C	保留	–	–	–
0x0040	FI DI 比率寄存器	US_FIDI	读 / 写	0x174
0x0044	错误数目寄存器	US_NER	只读	–
0x0048	保留	–	–	–
0x004C	IrDA 滤波寄存器	US_IF	读 / 写	0
0x5C-0xFC	保留	–	–	–
0x100-0x128	保留给 PDC 寄存器	–	–	–

USART 控制寄存器

寄存器名称： US_CR

访问类型： 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	RTSDIS	RTSEN	DTRDIS	DTREN

15	14	13	12	11	10	9	8
RETTO	RSTNACK	RSTIT	SENDA	STTTO	STPBRK	STTBRK	RSTSTA

7	6	5	4	3	2	1	0
TXDIS	TXEN	RXDIS	RXEN	RSTTX	RSTRX	–	–

- **RSTRX: 接收器复位**

0：无效。

1：接收器复位。

- **RSTTX: 发送器复位**

0：无效。

1：发送器复位。

- **RXEN: 接收器使能**

0：无效。

1：若 RXDIS 为 0，使能接收器。

- **RXDIS: 接收器禁用**

0：无效。

1：禁用接收器。

- **TXEN: 发送器使能**

0：无效。

1：若 TXDIS 为 0，使能发送器。

- **TXDIS: 发送器禁用**

0：无效。

1：禁用发送器。

- **RSTSTA: 状态位复位**

0：无效。

1：US_CSR 寄存器中状态位 PARE、 FRAME、 OVRE 与 RXBRK 复位。

- **STTBRK: 启动间断**

0：无效。

1：在 US_THR 中有字符且发送移位寄存器中字符已发送后，开始发送间断。若间断已发送则无效。

- **STPBRK: 停止间断**

0：无效。

1：最少一字符时间且发送高电平达 12 比特周期后停止间断发送。若间断已发送则无效。

• **STTTO: 启动超时**

0：无效。

1：在超时计数器计数前等待一个字符。

• **SENDA: 发送地址**

0：无效。

1：只适用于多点模式，发送写入 US_THR 的下一个字符并将地址位置位。

• **RSTIT: 迭代复位**

0：无效。

1：US_CSR 寄存器中 ITERATION 复位，若没有使能 ISO7816 则无效。

• **RSTNACK: 无应答复位**

0：无效。

1：US_CSR 寄存器中 NACK 复位。

• **RETTO: 重新超时**

0：无效。

1：超时重启。

• **DTREN: 数据终端就绪使能**

0：无效。

1：将 DTR 引脚驱动为 0。

• **DTRDIS: 数据终端就绪禁用**

0：无效。

1：将 DTR 引脚驱动为 1。

• **RTSEN: 发送请求使能**

0：无效。

1：将 RTS 引脚驱动为 0。

• **RTSDIS: 发送请求禁用**

0：无效。

1：将 RTS 引脚驱动为 1。

USART 模式寄存器

寄存器名称： US_MR

访问类型： 读 / 写

31	30	29	28	27	26	25	24
–	–	–	FILTER	–	MAX_ITERATION		

23	22	21	20	19	18	17	16
–	–	DSNACK	INACK	OVER	CLKO	MODE9	MSBF

15	14	13	12	11	10	9	8
CHMODE		NBSTOP		PAR			SYNC

7	6	5	4	3	2	1	0
CHRL		USCLKS		USART_MODE			

- **USART_MODE**

USART_MODE				USART 模式
0	0	0	0	普通
0	0	0	1	RS485
0	0	1	0	硬件握手
0	0	1	1	调制解调
0	1	0	0	IS07816 协议：T = 0
0	1	0	1	保留
0	1	1	0	IS07816 协议：T = 1
0	1	1	1	保留
1	0	0	0	IrDA
1	1	x	x	保留

- **USCLKS: 时钟选择**

USCLKS		时钟选择
0	0	MCK
0	1	MCK / DIV
1	0	保留
1	1	SCK

- **CHRL: 字符长度 .**

CHRL		字符长度
0	0	5 位
0	1	6 位
1	0	7 位
1	1	8 位



- **SYNC: 同步模式选择**

0：USART 工作在异步模式下。

1：USART 工作在同步模式下。

- **PAR: 检验类型**

PAR			检验类型
0	0	0	偶检验
0	0	1	奇检验
0	1	0	检验强制为 0 (空)
0	1	1	检验强制为 1 (标志)
1	0	x	无检验
1	1	x	多点模式

- **NBSTOP: 停止位数**

NBSTOP		异步 (SYNC = 0)	同步 (SYNC = 1)
0	0	1 个停止位	1 个停止位
0	1	1.5 个停止位	保留
1	0	2 个停止位	2 个停止位
1	1	保留	保留

- **CHMODE: 通道模式**

CHMODE		模式说明
0	0	普通模式
0	1	自动回应。接收器输入与 TXD 引脚连接。
1	0	本地回环。发送器输出与接收器输入连接。
1	1	远程回环。 RXD 引脚与 TXD 引脚连接。

- **MSBF: 位次序**

0：低位在先。

1：高位在先。

- **MODE9: 9 位字符长度**

0：CHRL 定义字符长度。

1：9 位字符长度。

- **CKLO: 时钟输出选择**

0：USART 不驱动 SCK 引脚。

1：若 USCLKS 未选择外部时钟 SCK， USART 驱动 SCK 引脚。

- **OVER: 重采样模式**

0：16 倍重采样。

1：8 倍重采样。

- **INACK: 抑制无应答**

0：产生 NACK。

1：不产生 NACK。

- **DSNACK: 连续 NACK 禁用**

0：一旦收到的字符出现检验错误， NACK 发送到 ISO 线上 (除非 INACK 置位)。

1: 连续检验错误达到 MAX_ITERATION 域给出的值。这些检验错误在 ISO 线上产生 NACK。一旦达到该值 ,ISO 线上不再发送另外的 NACK。出现 ITERATION 标志。

- **MAX_ITERATION**

定义模式 ISO7816，协议 T= 0 下最大迭代数。

- **FILTER: 红外接收线滤波器**

0：USART 不对接收线滤波。

1：USART 使用 3 采样滤波器 (1/16 位时钟) (2/3 多) 对接收线滤波。



USART 中断使能寄存器

寄存器名称 : US_IER

访问类型 : 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	CTSIC	DCDIC	DSRIC	RIIC
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	NACK	RXBUFF	TXBUFE	ITERATION	TXEMPTY	TIMEOUT
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PARE	FRAME	OVRE	ENDTX	ENDRX	RXBRK	TXRDY	RXRDY

- **RXRDY: RXRDY 中断使能**
- **TXRDY: TXRDY 中断使能**
- **RXBRK: 接收器间断中断使能**
- **ENDRX: 接收结束中断使能**
- **ENDTX: 发送结束中断使能**
- **OVRE: 溢出错误中断使能**
- **FRAME: 帧错误中断使能**
- **PARE: 奇偶错误中断使能**
- **TIMEOUT: 超时中断使能**
- **TXEMPTY: TXEMPTY 中断使能**
- **ITERATION: 迭代中断使能**
- **TXBUFE: 缓冲器空中断使能**
- **RXBUFF: 缓冲器满中断使能**
- **NACK: 无应答中断使能**
- **RIIC: 环指示器输入变化使能**
- **DSRIC: 数据设置就绪输入变化使能**
- **DCDIC: 数据载波检测输入变化中断使能**
- **CTSIC: 发送输入变化清除中断使能**

0 : 无效。

1 : 使能相应中断。

USART 中断禁用寄存器

寄存器名称： US_IDR

访问类型： 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	CTSIC	DCDIC	DSRIC	RIIC

15	14	13	12	11	10	9	8
–	–	NACK	RXBUFF	TXBUFE	ITERATION	TXEMPTY	TIMEOUT

7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	ENDTX	ENDRX	RXBRK	TXRDY	RXRDY

- **RXRDY: RXRDY 中断禁用**
- **TXRDY: TXRDY 中断禁用**
- **RXBRK: 接收器间断中断禁用**
- **ENDRX: 接收结束中断禁用**
- **ENDTX: 发送结束中断禁用**
- **OVRE: 溢出错误中断禁用**
- **FRAME: 帧错误中断禁用**
- **PARE: 奇偶错误中断禁用**
- **TIMEOUT: 超时中断禁用**
- **TXEMPTY: TXEMPTY 中断禁用**
- **ITERATION: 迭代中断禁用**
- **TXBUFE: 缓冲器空中断禁用**
- **RXBUFF: 缓冲器满中断禁用**
- **NACK: 无应答中断禁用**
- **RIIC: 环指示器输入变化禁用**
- **DSRIC: 数据设置就绪输入变化禁用**
- **DCDIC: 数据载波检测输入变化中断禁用**
- **CTSIC: 发送输入变化清除中断禁用**

0：无效。

1：禁用相应中断。



USART 中断屏蔽寄存器

寄存器名称 : US_IMR

访问类型 : 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	CTSIC	DCDIC	DSRIC	RIIC

15	14	13	12	11	10	9	8
–	–	NACK	RXBUFF	TXBUFE	ITERATION	TXEMPTY	TIMEOUT

7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	ENDTX	ENDRX	RXBRK	TXRDY	RXRDY

- **RXRDY: RXRDY 中断屏蔽**
- **TXRDY: TXRDY 中断屏蔽**
- **RXBRK: 接收器间断中断屏蔽**
- **ENDRX: 接收结束中断屏蔽**
- **ENDTX: 发送结束中断屏蔽**
- **OVRE: 溢出错误中断屏蔽**
- **FRAME: 帧错误中断屏蔽**
- **PARE: 奇偶错误中断屏蔽**
- **TIMEOUT: 超时中断屏蔽**
- **TXEMPTY: TXEMPTY 中断屏蔽**
- **ITERATION: 迭代中断屏蔽**
- **TXBUFE: 缓冲器空中断屏蔽**
- **RXBUFF: 缓冲器满中断屏蔽**
- **NACK: 无应答中断屏蔽**
- **RIIC: 环指示器输入变化屏蔽**
- **DSRIC: 数据设置就绪输入变化屏蔽**
- **DCDIC: 数据载波检测输入变化中断屏蔽**
- **CTSIC: 发送输入变化清除中断屏蔽**

0 : 相应中断禁用。

1 : 相应中断使能。

USART 通道状态寄存器

寄存器名称 : US_CSR

访问类型 : 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
CTS	DCD	DSR	RI	CTSIC	DCDIC	DSRIC	RIIC

15	14	13	12	11	10	9	8
–	–	NACK	RXBUFF	TXBUFE	ITERATION	TXEMPTY	TIMEOUT

7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	ENDTX	ENDRX	RXBRK	TXRDY	RXRDY

- **RXRDY: 接收器就绪**

0 : 上次读 US_RHR 后未接收到完整的字符或接收器禁用。若接收器禁用时正在接收字符，当接收器使能时 RXRDY 变为 1。

1 : 上次读 US_RHR 后至少收到一个完整的字符，且 US_RHR 未被读取。

- **TXRDY: 发送器就绪**

0 : US_THR 中字符等待发送到发送移位寄存器，或请求 STTBRK 命令，或发送器禁用。一旦发送器使能，TXRDY 变为 1。

1 : US_THR 中无字符。

- **RXBRK: 间断接收 / 间断结束**

0 : 上次 RSTSTA 后未检测到间断接收或间断结束。

1 : 上次 RSTSTA 后检测到间断接收或间断结束。

- **ENDRX: 接收器传输结束**

0 : 接收 PDC 通道的传输结束信号无效。

1 : 接收 PDC 通道的传输结束信号有效。

- **ENDTX: 发送器传输结束**

0 : 发送 PDC 通道的传输结束信号无效。

1 : 发送 PDC 通道的传输结束信号有效。

- **OVRE: 溢出错误**

0 : 上次 RSTSTA 后未出现溢出错误。

1 : 上次 RSTSTA 后至少出现一次溢出错误。

- **FRAME: 帧错误**

0 : 上次 RSTSTA 后未检测到停止位。

1 : 上次 RSTSTA 后至少检测到一个停止位。

- **PARE: 检验错误**

0 : 上次 RSTSTA 后未检测到检验错误。

1 : 上次 RSTSTA 后至少检测到一次检验错误。

- **TIMEOUT: 接收器超时**

0 : 上次启动超时命令后无超时或超时寄存器为 0。

1 : 上次启动超时命令后有超时。



- **TXEMPTY: 发送器空**

0：US_THR 与发送移位寄存器中均无字符，或发送器禁用。

1：US_THR 与发送移位寄存器中至少有一个字符。

- **ITERATION: 达到最大重复数**

0：上次 RSIT 后还未达到最大重复数。

1：上次 RSIT 后已达到最大重复数。

- **TXBUFE: 发送缓冲器空**

0：发送 PDC 通道缓冲器空信号无效。

1：发送 PDC 通道缓冲器空信号有效。

- **RXBUFF: 接收缓冲器满**

0：接收 PDC 通道缓冲器满信号无效。

1：接收 PDC 通道缓冲器满信号有效。

- **NACK: 无应答**

0：上次 RSTNACK 后未检测到无应答。

1：上次 RSTNACK 后至少检测到一次无应答。

- **RIIC: 环指示器输入变化标志**

0：上次读 US_CSR 后在 RI 引脚上未检测到输入变化。

1：上次读 US_CSR 后在 RI 引脚上至少检测到一次输入变化。

- **DSRIC: 数据设置就绪输入变化标志**

0：上次读 US_CSR 后在 DSR 引脚上未检测到输入变化。

1：上次读 US_CSR 后在 DSR 引脚上至少检测到一次输入变化。

- **DCDIC: 数据载波检测输入变化标志**

0：上次读 US_CSR 后在 DCD 引脚上未检测到输入变化。

1：上次读 US_CSR 后在 DCD 引脚上至少检测到一次输入变化。

- **CTSIC: 发送输入变化清除标志**

0：上次读 US_CSR 后在 CTS 引脚上未检测到输入变化。

1：上次读 US_CSR 后在 CTS 引脚上至少检测到一次输入变化。

- **RI: RI 输入映象**

0：RI 为 0。

1：RI 为 1。

- **DSR: DSR 输入映象**

0：DSR 为 0。

1：DSR 为 1。

- **DCD: DCD 输入映象**

0：DCD 为 0。

1：DCD 为 1。

- **CTS: CTS 输入映象**

0：CTS 为 0。

1：CTS 为 1。

USART 接收保持寄存器

寄存器名称 : US_RHR

访问类型 : 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	RXCHR

7	6	5	4	3	2	1	0
RXCHR							

- **RXCHR: 收到的字符**

若 RXRDY 置位，为接收到的最后一个字符。

USART 发送保持寄存器

寄存器名称 : US_THR

访问类型 : 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	TXCHR

7	6	5	4	3	2	1	0
TXCHR							

- **TXCHR: 将发送的字符**

若 TXRDY 未置位，则为下一个要发送的字符。

USART 波特率发送器寄存器

寄存器名称 : US_BRGR

访问类型 : 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
CD							

7	6	5	4	3	2	1	0
CD							

- **CD: 时钟分频器**

CD	USART_MODE ≠ ISO7816			USART_MODE = ISO7816
	SYNC = 0		SYNC = 1	
	OVER = 0	OVER = 1		
0	波特率时钟禁用			
1 到 65535	波特率 = 选定时钟 /16/CD	波特率 = 选定时钟 /8/CD	波特率 = 选定时钟 /CD	波特率 = 选定时钟 /CD/FI_DI_RATIO

USART 接收器超时寄存器

寄存器名称 : US_RTOR

访问类型 : 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
TO							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TO							

- **TO: 超时值**

0 : 接收器超时禁用。

1 - 65535 : 接收器超时使能且超时延迟为 TO x 比特周期。

USART 发送器时间保障寄存器

寄存器名称 : US_TTGR

访问类型 : 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TG							

- **TG: 时间保障值**

0 : 发送器时间保障禁用。

1 - 255 : 发送器时间保障使能且时间保障延迟为 TG x 比特周期。



USART FI DI 比率寄存器

寄存器名称 : US_FIDI

访问类型 : 读 / 写

复位值 :: 0x174

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	FI_DI_RATIO		

7	6	5	4	3	2	1	0
FI_DI_RATIO							

- **FI_DI_RATIO: FI 与 DI 比值**

0 : 若选择 ISO7816 模式，波特率发生器不产生信号。

1 - 2047 : 若选择 ISO7816 模式，波特率为 SCK 时钟与 FI_DI_RATIO 相除后的值。

USART 错误数目寄存器

寄存器名称 : US_NER

访问类型 : 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
NB_ERRORS							

- **NB_ERRORS: 错误数目**

ISO7816 传输中错误总数。当被读取时，该寄存器自动清零。

USART IrDA 滤波器寄存器

寄存器名称： US_IF

访问类型： 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
IRDA_FILTER							

- **IRDA_FILTER: IrDA 滤波器**

设置 IrDA 解调器滤波器。



同步串行控制器 (SSC)

概述

Atmel 同步串行控制器 (SSC) 提供与外部器件的同步通信。它支持许多用于音频及电信应用中常用的串行同步通信协议，如 I2S、短帧同步、长帧同步等。

SSC 包含独立的接收器、发送器及一个时钟分频器。每个发送器及接收器有三个接口：针对数据的 TD/RD 信号、针对时钟的 TK/RK 信号及针对帧同步的 TF/RF 信号。最多可传输 16 个 32 位数据。可编程设定为自动启动或在帧同步信号检测到不同事件时启动。

SSC的可编程高电平及两个32位专用PDC 通道 , 可在没有处理器干涉的情况下进行连续的高速率数据传输。

由于与两个 PDC 通道连接， SSC 可在低处理器开销的情况下与下列器件连接：

- 主机或从机模式下的 CODEC
- 专用串行接口的 DAC，特别是 I2S
- 磁卡阅读器

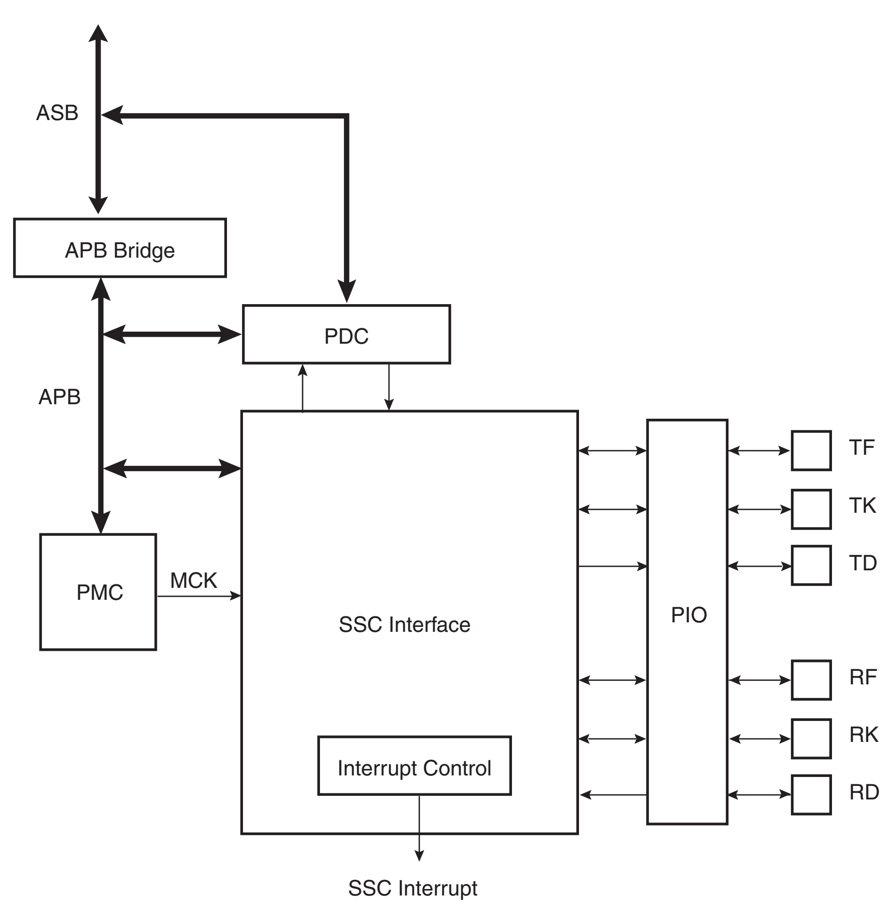
SSC 特性如下：

- 提供用于音频与电信领域的串行同步通信链接
- 包含一个独立的接收器与发送器及通用时钟分频器
- 与两个 PDC 通道连接 (DMA 访问) 以减少处理器开销
- 提供可配置帧同步与数据长度
- 接收器与发送器可编程为自动启动或检测帧同步信号的不同事件
- 发送器与接收器包括一个数据信号、一个时钟信号及一个帧同步信号



方框图

Figure 202. 方框图



应用框图

Figure 203. 应用框图

OS or RTOS Driver		Power Management	Interrupt Management		Test Management
SSC					
Serial AUDIO	Codec	Time Slot Management		Frame Management	Line Interface

引脚名称列表

Table 85. I/O 线说明

引脚名称	引脚说明	类型
RF	接收器帧同步	输入 / 输出
RK	接收器时钟	输入 / 输出
RD	接收器数据	输入
TF	发送器帧同步	输入 / 输出
TK	发送器时钟	输入 / 输出
TD	发送器数据	输出

附属产品

I/O 线

与外设引脚连接可与 PIO 线复用。

使用 SSC 接收器前， PIO 控制器必须将专用的 SSC 接收器 I/O 线配置为 SSC 外设模式。

使用 SSC 发送器前， PIO 控制器必须将专用的 SSC 发送器 I/O 线配置为 SSC 外设模式。

电源管理

SSC时钟不连续。SSC接口可由电源管理控制器(PMC)计时，因此必须先配置PMC以使能SSC时钟。

中断

SSC接口有与高级中断控制器(AIC)连接的中断线。处理中断请求须在配置SSC前对AIC编程。

所有 SSC 中断可通过配置 SSC 中断屏蔽寄存器来使能 / 禁用。每个挂起与未屏蔽的 SSC 中断将出现在SSC中断线上。SSC中断服务子程序可通过读取SSC中断状态寄存器来得到中断源。



功能说明

该节包括以下功能说明 : SSC 功能块、时钟管理、数据格式、启动、发送器、接收器及帧同步。

接收器与发送器独立工作。但可通过编程使接收器使用发送时钟或当发送启动时开始数据传输 , 以使它们同步工作。也可通过编程使发送器使用接收时钟或当接收启动时开始数据传输 , 实现二者同步工作。发送器与接收器时钟可由 TK 或 RK 引脚提供。使得 SSC 支持多从机模式数据传输。TK 与 RK 引脚的最大时钟速率为主机时钟的 2 分频。每级时钟至少要稳定两个主机时钟。

Figure 204. SSC 功能框图

时钟管理

发送器时钟可由以下产生：
- TK I/O 上接收的外部时钟
- 接收器时钟
- 内部时钟分频器

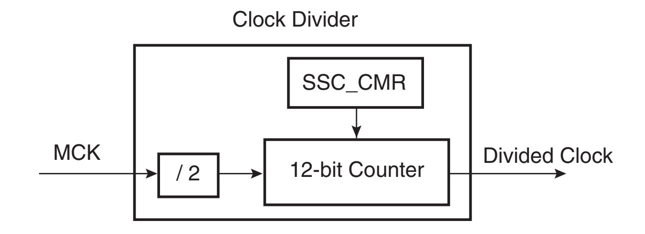
接收器器时钟可由以下产生：
- RK I/O 上接收的外部时钟
- 发送器时钟
- 内部时钟分频器

此外，发送器可在 TK I/O 上产生外部时钟，接收器可在 RK I/O 上产生外部时钟。

因此， SSC 支持多主机与从机模式数据传输。

时钟分频器

Figure 205. 时钟分频框图



主机时钟分频器由时钟模式寄存器SSC_CMR的12位域DIV计数器及比较器(其最大值为4095)确定，最高可对主机时钟 8190 分频。分频后时钟提供给接收器与发送器。当该域编程为 0，时钟分频器不使用并保持无效。

当 DIV 值大于或等于 1，分频后时钟频率为主机时钟 2 倍 DIV 分频后的值。每级分频时钟电平持续时间为主机时钟与 DIV 的乘积。这保证无论 DIV 为奇数还是偶数，分频后时钟占空比为 50% 。

Figure 206. 分频时钟产生

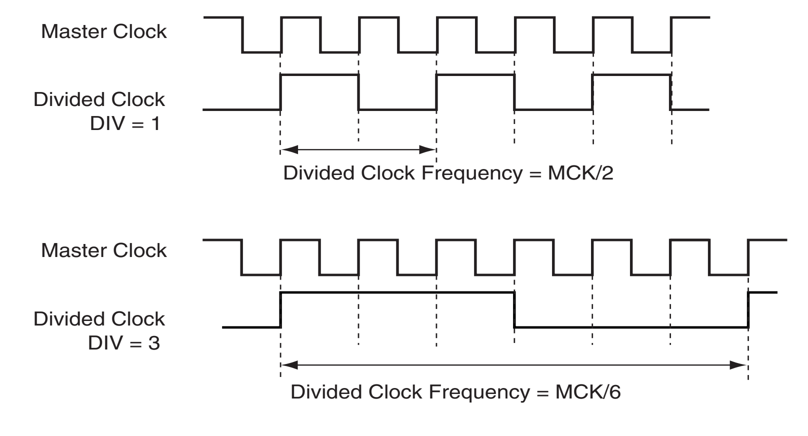


Table 86. 比特率

最大	最小
MCK / 2	MCK / 8190

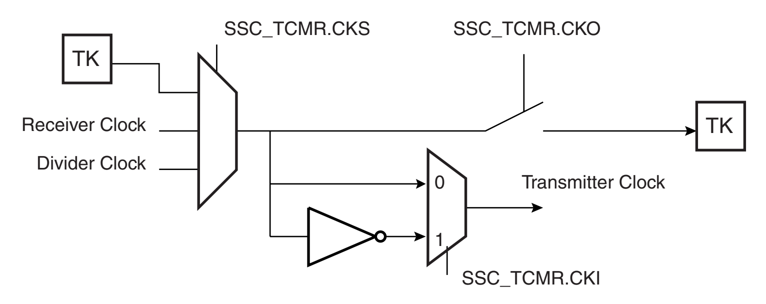
发送器时钟管理

发送器时钟来自接收器时钟或分频器时钟或 TK I/O 上的外部时钟。发送器时钟由 SSC_TCMR (发送时钟模式寄存器) 的 CKS 域选择。发送时钟可通过 SSC_TCMR 的 CKI 位反转。



发送器可连续驱动 TK I/O 或受限于实际数据传输。时钟输出由 SSC_TCMR 寄存器配置。发送时钟反转 (CKI) 位对时钟输出无效。对 TCMR 寄存器编程选择 TK 引脚 (CKS 域) 但同时连续发送时钟 (CKO 域) 可能引起不可预见的结果。

Figure 207. 发送器时钟管理

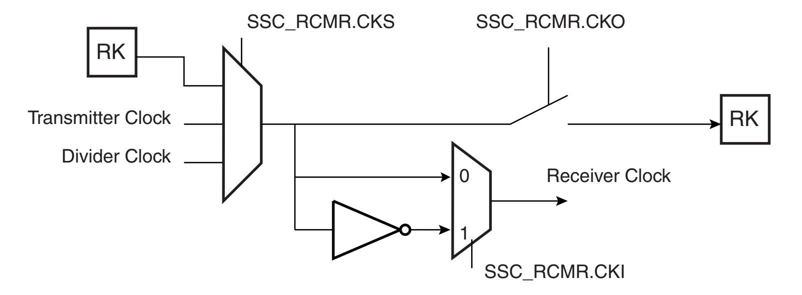


接收器时钟管理

接收器时钟来自发送器时钟或分频器时钟或 RK I/O 上的外部时钟。接收时钟由 SSC_RCMR (接收时钟模式寄存器) 的 CKS 域选择。接收时钟可通过 SSC_RCMR 的 CKI 位反转。

接收器可连续驱动 RK I/O 或受限于实际数据传输。时钟输出由 SSC_RCMR 寄存器配置。接收时钟反转 (CKI) 位对时钟输出无效。对 RCMR 寄存器编程选择 RK 引脚 (CKS 域) 但同时连续接收时钟 (CKO 域) 可能引起不可预见的结果。

Figure 208. 接收器时钟管理



发送器操作

发送帧由启动事件触发，并可在数据发送前加入同步数据。

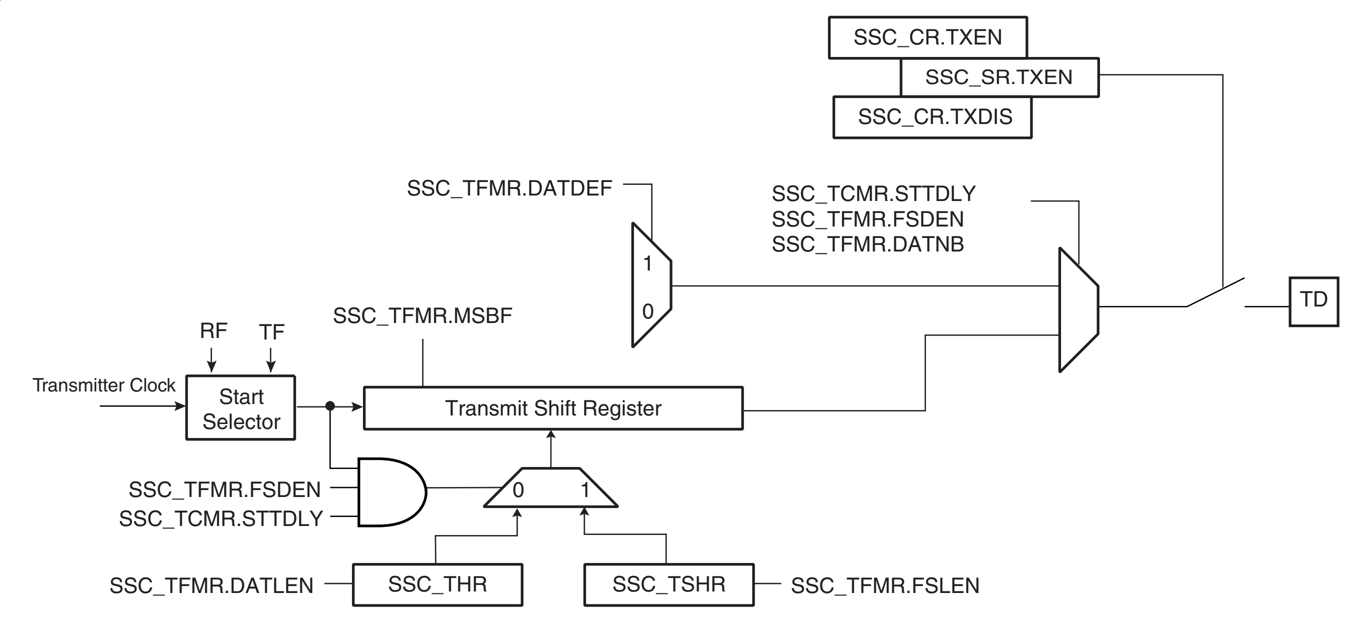
启动事件通过时钟发送时钟模式寄存器 (SSC_TCMR) 来配置，见 See " 启动 " on page 442.。

帧同步通过发送帧模式寄存器 (SSC_TFMR) 来配置，见 See " 帧同步 " on page 444.。

发送数据时，发送器使用发送器时钟信号作为移位寄存器时钟并在 SSC_TCMR 中选择启动模式。根据应用将数据写入 SSC_THR 寄存器，然后根据选择的数据格式将数据传输到移位寄存器中。

当 SSC_THR 与发送移位寄存器均为空时，SSC_SR 中状态标志 TXEMPTY 置位。当发送保持寄存器传输到移位寄存器时， SSC_SR 中 TXRDY 位置位，新数据可载入保持寄存器中。

Figure 209. 发送器框图



接收器操作

接收器帧由启动事件触发，并可在数据发送前加入同步数据。

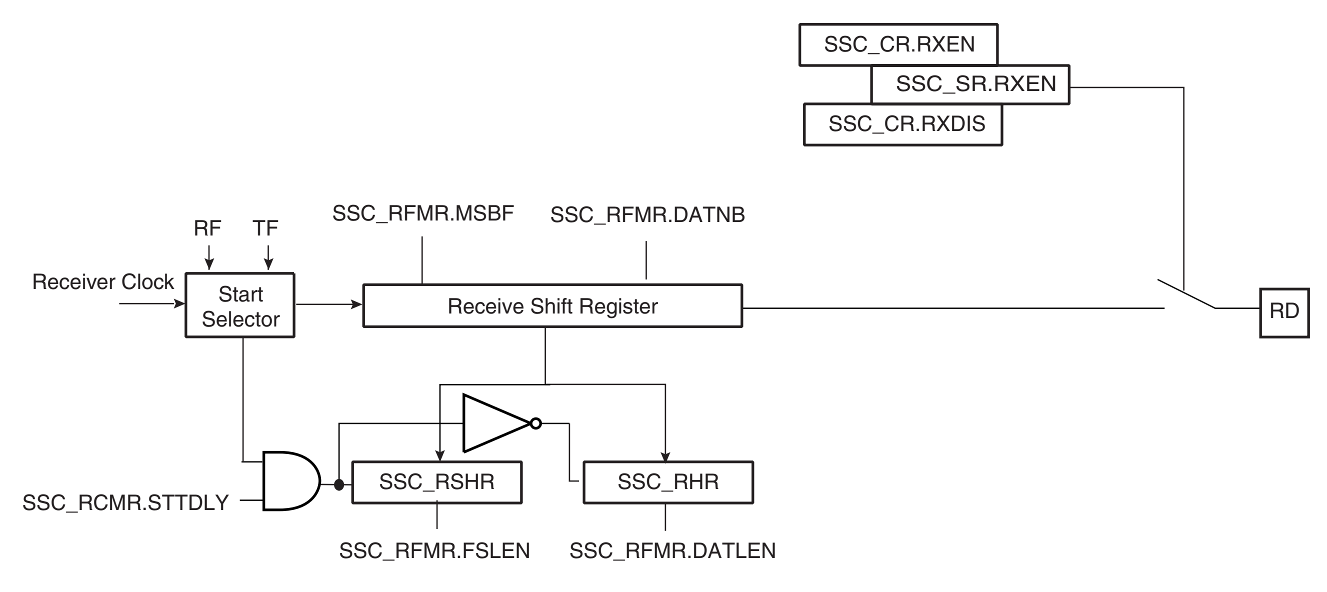
启动事件通过时钟接收时钟模式寄存器 (SSC_RCMR) 来配置，见 See “ 启动 ” on page 442.。

帧同步通过接收帧模式寄存器 (SSC_RFMR) 来配置，见 See “ 帧同步 ” on page 444.。

接收数据时，接收器使用接收器时钟信号作为移位寄存器时钟并在 SSC_RCMR 中选择启动模式。根据选择的数据格式将数据传输到移位寄存器中。

当接收移位寄存器满， SSC 将数据送入保持寄存器， SSC_SR 中状态标志 RXRDY 置位，若在读 RHR 寄存器前有其它传输出现， SSC_SR 中 OVERUN 位置位，且接收器移位寄存器传输到 RHR 寄存器。

Figure 210. 接收器框图



启动

发送器与接收器可编程设定为当事件出现时开始工作，分别设置 SSC_TCMR 中的发送启动选择 (START) 域及 SSC_RCMR 中的接收启动选择 (START) 域。

在以下情况中，启动事件可独立编程：

- 连续。这种情况下，一旦 SSC_THR 中写入数据，即开始发送，并且在接收器时接收启动。
- 与发送器 / 接收器同步
- 检测到 TK/RK 的上升 / 下降沿
- 检测到 TK/RK 的高 / 低电平
- 检测到 TK/RK 的电平变化或跳变沿

可在发送 / 接收时钟寄存器 (RCMR/TCMR) 的任意边沿以同样的方法编程设置。因此，可在 TF (发) 或 RF (接收) 启动。

通过发送 / 接收帧模式寄存器 (TFMR/RFMR) 的 FSOS 域，检测 TF/RF 输入 / 输出。

帧同步必须在与其相关输出上产生。

Figure 211. 发送启动模式

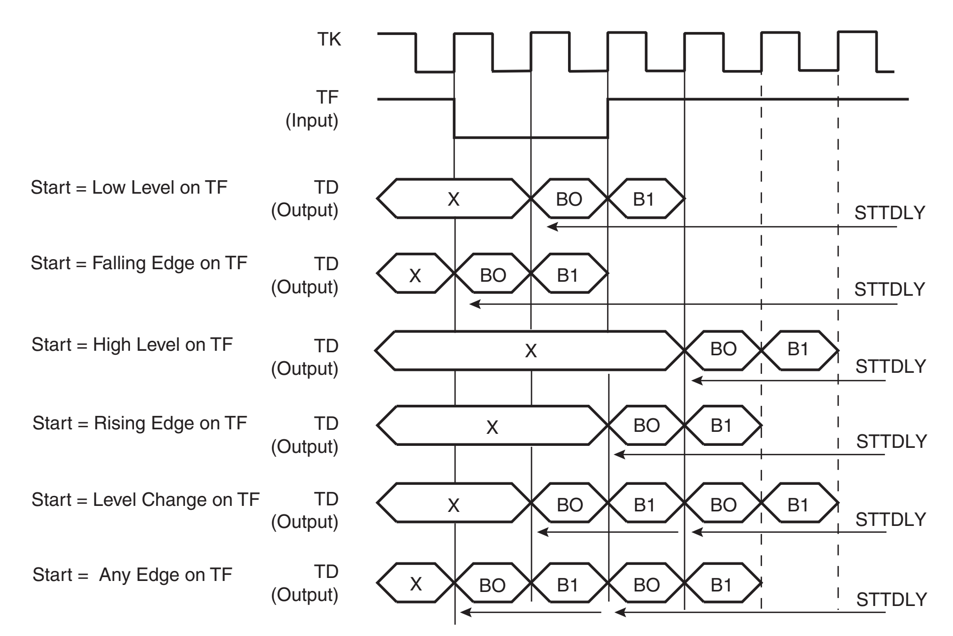
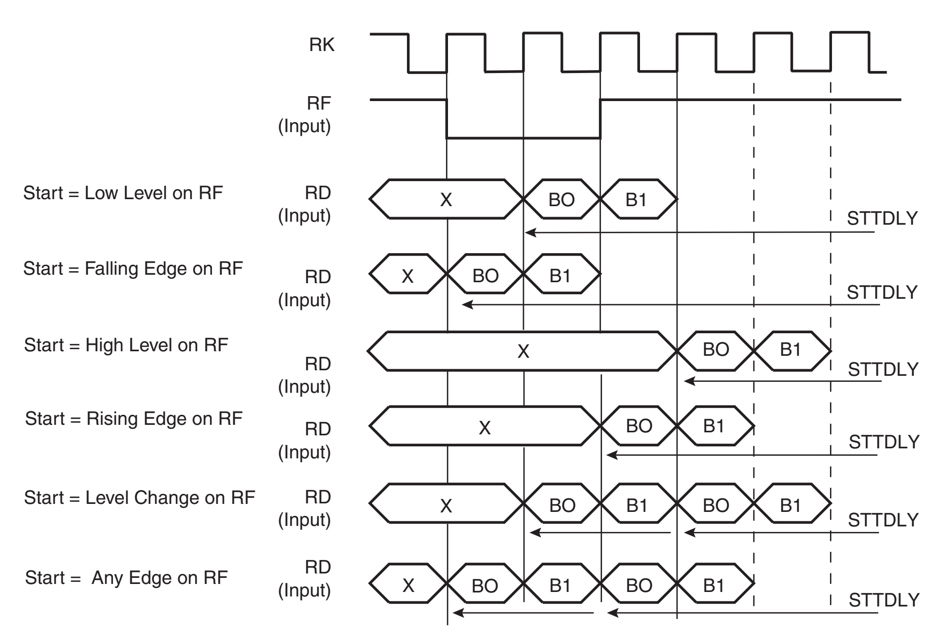


Figure 212. 接收脉冲 / 边沿启动模式



帧同步

发送器与接收器帧同步引脚 TF 与 RF 可编程产生不同类型帧同步信号。发送帧模式寄存器 (SSC_RFMR) 中的及发送帧模式寄存器 (SSC_TFMR) 中的帧同步输出选择 (FSOS) 域用来选择所需波形。

- 支持数据传输中的可编程高低电平。
- 支持数据传输启动前高电平或电平切换。

若选择脉冲波形， SSC_RFMR 与 SSC_TFMR 中帧同步长度 (FSLEN) 域对脉冲长度编程，由 1 比特时间到 16 比特时间。

接收与发送帧同步脉冲周期可通过 SSC_RCMR 与 SSC_TCMR 的周期分频器选择 (PERIOD) 域编程设定。

帧同步数据

帧同步数据在帧同步信号中发送或接收一个特定的标记。

帧同步信号中，接收器可对 RD 线采样并在接收同步保持寄存器中保存数据，发送器可将发送同步保持寄存器发送移位寄存器中。在由 SSC_RFMR/SSC_TFMR 中 FSLEN 域编程设定的帧同步信号中可对数据长度进行采样或移出。

考虑接收帧同步数据操作，若帧同步长度等于或小于启动事件与实际数据接收间的延迟时，通过接收移位寄存器的接收同步保持寄存器执行数据采样。

只有当 SSC_TFMR 寄存器中帧同步数据使能 (FSDEN) 位置位时，执行帧同步操作。若帧同步长度等于或小于启动事件与实际数据发送间的延迟时，普通发送优先且将发送同步保持寄存器中的数据送到发送寄存器中，然后再移出。

帧同步边沿检测

帧同步边沿检测由 SSC_RFMR/SSC_TFMR 中 FSEDGE 域编程设定。将 SSC 中状态寄存器 (SSC_SR) 的 RXSYN/TXSYN 标志设定为帧同步边沿检测 (RF/TF 信号)。

数据格式

发送器与接收器的数据帧格式基本上由发送器帧模式寄存器 (SSC_TFMR) 及接收器帧模式寄存器 (SSC_RFMR) 编程设定。任一情况下，用户可分别选择：

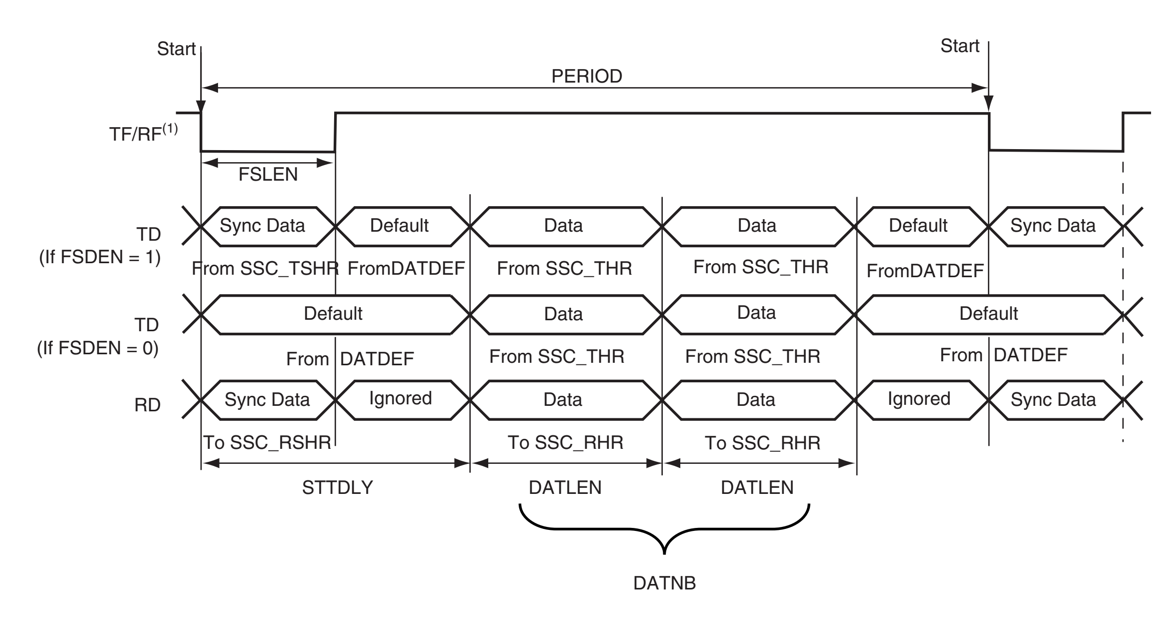
- 启动数据传输事件 (START)。
- 启动时间到首个数据位间位周期延迟数 (STTDLY)。
- 数据长度 (DATLEN)。
- 每次启动事件传输的数据数 (DATNB)。
- 每次启动事件同步传输长度 (FSLEN)。
- 比特意义：高位或低位在先 (MSBF)。

  此外，没有进行数据传输时，发送器可用来发送同步并选择 TD 引脚驱动电平。分别由 SSC_TFMR 中帧同步数据使能 (FSDEN) 位及数据默认值 (DATDEF) 位实现。

Table 87. 数据帧寄存器

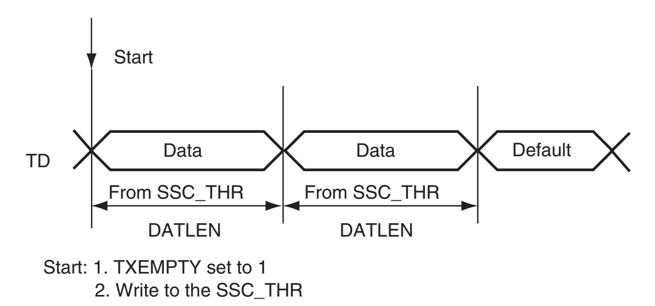
发送器	接收器	域	长度	注释
SSC_TFMR	SSC_RFMR	DATLEN	达到 32	字长
SSC_TFMR	SSC_RFMR	DATNB	达到 16	发送器帧字数
SSC_TFMR	SSC_RFMR	MSBF		1 高位在先
SSC_TFMR	SSC_RFMR	FSLEN	达到 16	同步数据寄存器大小
SSC_TFMR		DATDEF	0 或 1	数据默认值结束
SSC_TFMR		FSDEN		使能发送 SSC_TSHR
SSC_TCMR	SSC_RCMR	PERIOD	达到 512	帧大小
SSC_TCMR	SSC_RCMR	STTDLY	达到 255	发送启动延迟大小

Figure 213. 边沿 / 脉冲启动模式下发送与接收帧格式



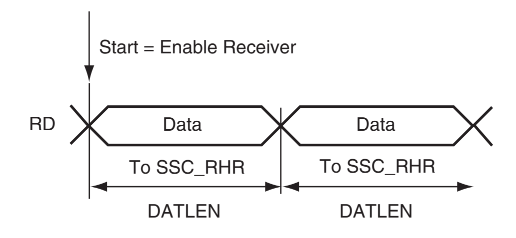
Note: 1. TF/RF 输入下降沿示例。

Figure 214. 连续模式下发送帧格式



Note: 1. STTDLY 设置为 0。本例中，SSC_THR 载入两次。FSDEN 值对发送无效。连续模式下不能输出同步数据。

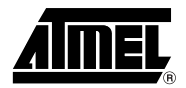
Figure 215. 连续模式下接收帧格式



Note: 1. STTDLY 设置为 0。

循环模式

接收器可编程为接收发送器发送的数据。通过设定 SSC_RFMR 中循环模式 (LOOP) 位实现。此时， RD 与 TD 连接， RF 与 TF 连接， RK 与 TK 连接。

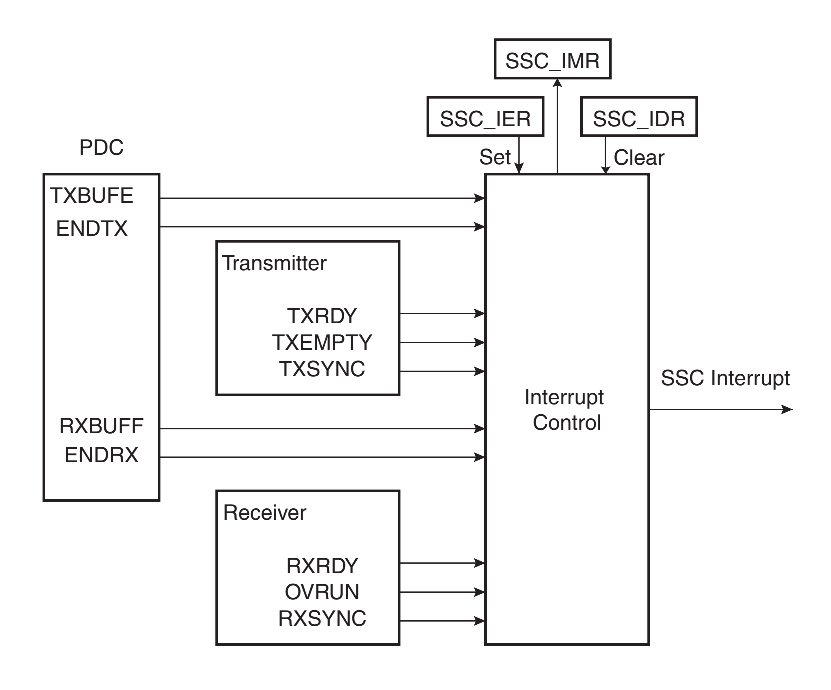


中断

SSC_SR 中大部分位在中断管理寄存器中有对应位。

SSC控制器可编程为当其检测到事件时产生中断。中断控制通过写SSC_IER (中断使能寄存器) 及 SSC_IDR (中断禁用寄存器) 来实现 ,二者通过在 SSC_IMR (中断屏蔽寄存器) 的相应位置位或清零来分别使能与禁用相应中断，SSC_IMR 通过在与 AIC 连接的 SSC 中断线上出现来产生中断。

Figure 216. 中断框图



SSC 应用示例

SSC 支持用于音频或高速串行链接的几种串行通信模式。下面给出一些标准应用。所有 SSC 支持的串行链接应用在此处未列出。

Figure 217. 音频应用框图

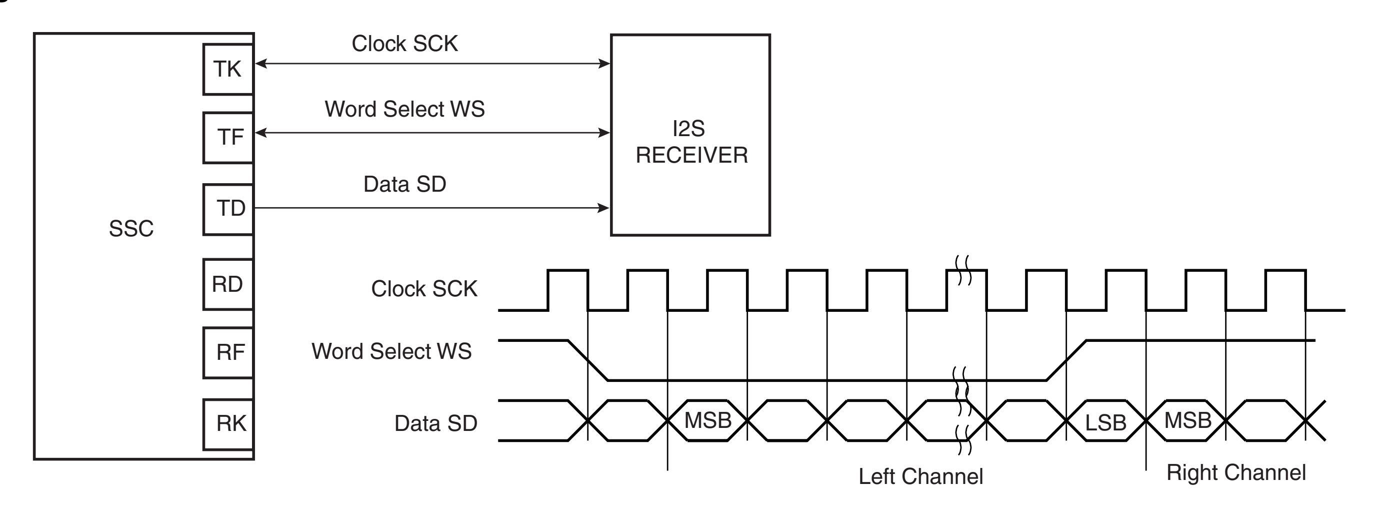


Figure 218. 编译码器应用框图

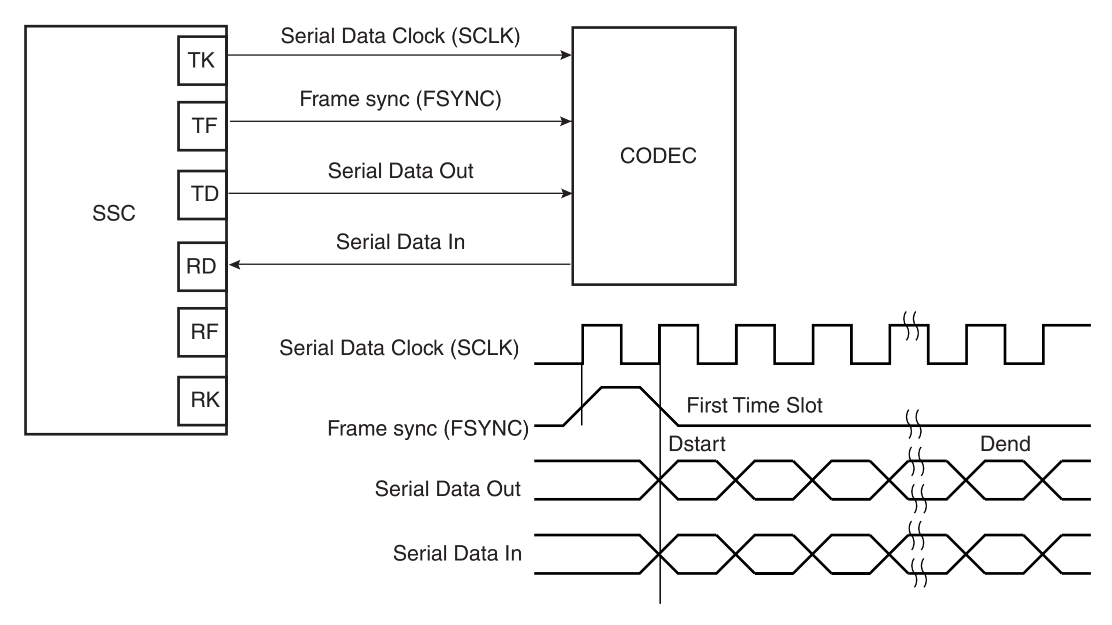
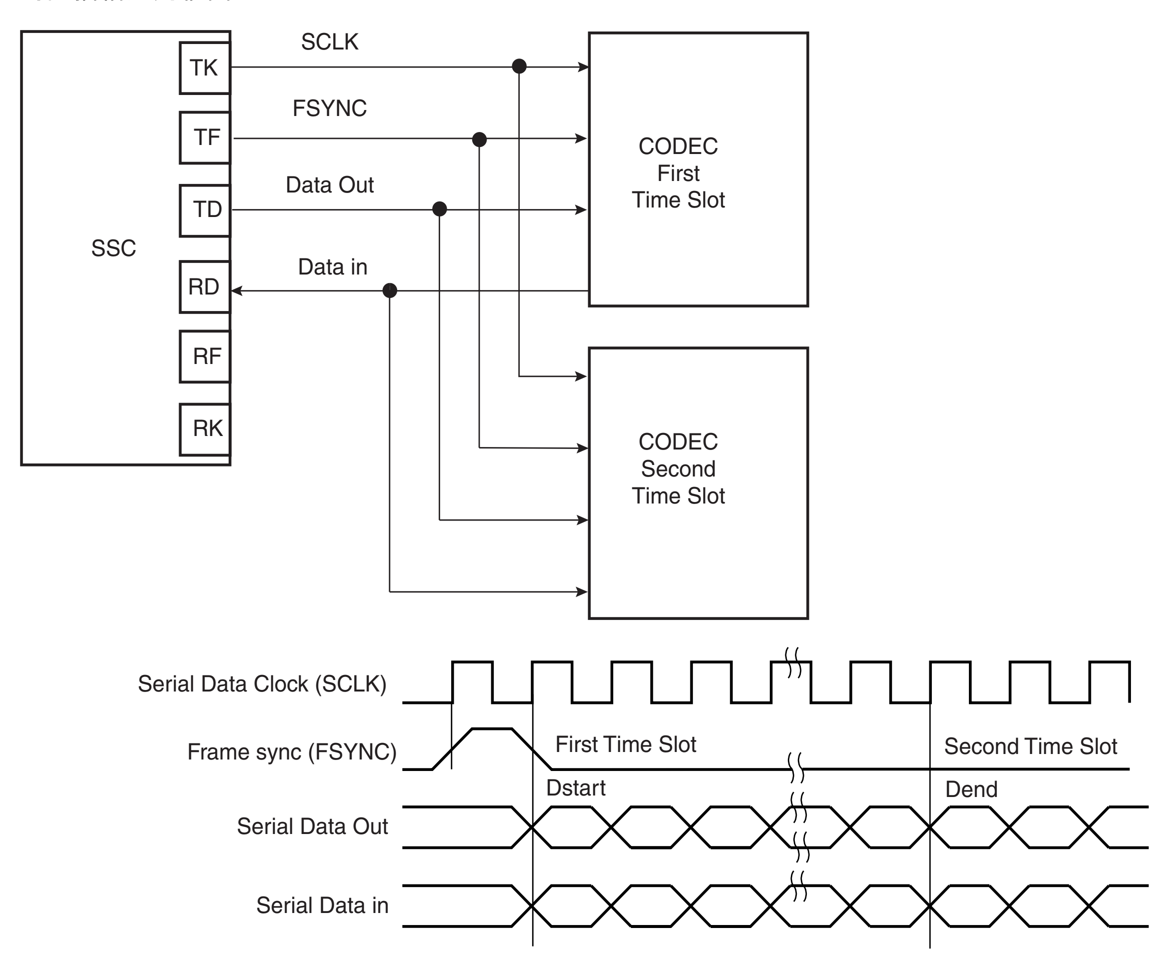


Figure 219. 时间插槽应用框图





同步串行控制器 (SSC) 用户接口

Table 88. SSC 寄存器映射

偏移	寄存器	寄存器名称	访问类型	复位值
0x0	控制寄存器	SSC_CR	写	–
0x4	时钟模式寄存器	SSC_CMR	读 / 写	0x0
0x8	保留	–	–	–
0xC	保留	–	–	–
0x10	接收时钟模式寄存器	SSC_RCMR	读 / 写	0x0
0x14	接收帧模式寄存器	SSC_RFMR	读 / 写	0x0
0x18	发送时钟模式寄存器	SSC_TCMR	读 / 写	0x0
0x1C	发送帧模式寄存器	SSC_TFMR	读 / 写	0x0
0x20	接收保持寄存器	SSC_RHR	读	0x0
0x24	发送保持寄存器	SSC_THR	写	–
0x28	保留	–	–	–
0x2C	保留	–	–	–
0x30	接收同步保持寄存器	SSC_RSHR	读	0x0
0x34	发送同步保持寄存器	SSC_TSHR	读 / 写	0x0
0x38	保留	–	–	–
0x3C	保留	–	–	–
0x40	状态寄存器	SSC_SR	读	0x000000CC
0x44	中断使能寄存器	SSC_IER	写	–
0x48	中断禁用寄存器	SSC_IDR	写	–
0x4C	中断屏蔽寄存器	SSC_IMR	读	0x0
0x50-0xFF	保留	–	–	–
0x100- 0x124	为外设数据控制器 (PDC) 保留	–	–	–

SSC 控制寄存器

寄存器名称 : SSC_CR

访问类型 : 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
SWRST	–	–	–	–	–	TXDIS	TXEN
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	–	–	–	–	RXDIS	RXEN

- **RXEN: 接收使能**

0 : 无效。

1 : 若 RXDIS 未置位，使能数据接收 [1]。

- **RXDIS: 接收禁用**

0 : 无效。

1 : 禁用数据接收 [1]。

- **TXEN: 发送使能**

0 : 无效。

1 : 若 TXDIS 未置位，使能数据发送 [1]。

- **TXDIS: 发送禁用**

0 : 无效。

1 : 禁用数据发送 [1]。

- **SWRST: 软件复位**

0 : 无效。

1 : 执行软件复位。比 SSC_CR 其它位优先。

Note: 1. 只有数据管理受影响。

SSC 时钟模式寄存器

寄存器名称 : SSC_CMR

访问类型 : 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	DIV			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DIV							

- **DIV: 时钟分频器**

0 : 时钟分频器无效。

其它任意值 : 分频时钟等于主机时钟除以 2 倍的 DIV。最大位速率为 MCK/2 ；最小位速率为 MCK/2 x 4095 = MCK/8190。



SSC 接收时钟模式寄存器

寄存器名称： SSC_RCMR

访问类型： 读 / 写

31	30	29	28	27	26	25	24
PERIOD							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
STTDLY							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	START			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	CKI	CKO			CKS	

- **CKS: 接收时钟选择**

CKS	接收时钟选择
0x0	分频时钟
0x1	TK 时钟信号
0x2	RK 引脚
0x3	保留

- **CKO: 接收时钟输出模式选择**

CKO	接收时钟输出模式	RK 引脚
0x0	无	单输入
0x1	连续接收时钟	输出
0x2-0x7	保留	

- **CKI: 接收时钟反转**

0：数据及帧同步信号在接收时钟下降沿采样。

1：数据及帧同步信号在接收时钟下降沿移出。

CKI 不影响 RK 输出时钟信号。

- **START: 接收启动选择**

启动	接收启动
0x0	连续，一旦接收器使能，在前一数据传输结束后立即开始
0x1	发送启动
0x2	检测 RF 输入低电平
0x3	检测 RF 输入高电平
0x4	检测 RF 输入下降沿
0x5	检测 RF 输入上升沿
0x6	检测 RF 输入电平变化
0x7	检测 RF 输入任意边沿
0x8-0xF	保留

- **STTDLY: 接收启动延迟**

若 STTDLY 不为 0，在启动事件与实际开始接收间插入一个 STTDLY 时钟周期的延迟。当接收器编程与发送器同步时，延迟依然有效。

请注意 : 设置 STTDLY 时需非常小心。若必须设置 STTDLY，应考虑与其相关的 TAG (接收同步数据) 接收。

- **PERIOD: 接收周期分频器选择**

该域选择应用于所选时钟的分频器，以产生一个新的帧同步信号。若为 0，不产生 PERIOD 信号；若不为 0，每 2 x (PERIOD+1) 个接收时钟产生一个 PERIOD 信号。



SSC 接收帧模式寄存器

寄存器名称： SSC_RFMR

访问类型： 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	FSEDGE
23	22	21	20	19	18	17	16
–	FSOS			FSLEN			
15	14	13	12	11	10	9	8
–	–	–	–	DATNB			
7	6	5	4	3	2	1	0
MSBF	–	LOOP	DATLEN				

- **DATLEN: 数据长度**

不支持 0x0。 DATLEN 值可设定在 0x1 到 0x1F 间。

位串包含 DATLEN + 1 数据位。此外，它定义了接收器 PDC 执行的传输大小。

若 DATLEN 小于或等于 7，数据以字节传输；若 DATLEN 在 8 到 15 间 (包括 15)，数据以半字传输；其它值时，数据以 32 位字传输。

- **LOOP: 循环模式**

0：普通工作模式。

1：RD 由 TD 驱动， RF 由 TF 驱动， TK 驱动 RK。

- **MSBF: 高位在先**

0：首先对数据寄存器中的低位采样。

1：首先对数据寄存器中的高位采样。

- **DATNB: 每帧数据数**

该域定义每次发送开始后接收的数据字数。若为 0，只传输 1 个数据字，最多可传输 16 个数据字。

- **FSLEN: 接收帧同步长度**

该域定义接收帧同步信号长度及采样并保存于接收同步数据寄存器中的数据位数。只有当 FSOS 在正脉冲或负脉冲时设置才有效。

- **FSOS: 接收帧同步输出选择**

FSOS	选定的接收帧同步信号	RF 引脚
0x0	无	单输入
0x1	负脉冲	输出
0x2	正脉冲	输出
0x3	数据传输时拉低	输出
0x4	数据传输时拉高	输出
0x5	每次数据传输启动时切换	输出
0x6-0x7	保留	未定义

- **FSEDGE: 帧同步边沿检测**

确定帧同步边沿，在 SSC 状态寄存器的 RXSYN 中设置。

FSEDGE	帧同步边沿检测
0x0	正沿检测
0x1	负沿检测

SSC 发送时钟模式寄存器

寄存器名称： SSC_TCMR

访问类型： 读 / 写

31	30	29	28	27	26	25	24
PERIOD							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
STTDLY							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	START			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	CKI	CKO			CKS	

- **CKS: 发送时钟选择**

CKS	选定的发送时钟
0x0	分频时钟
0x1	RK 时钟信号
0x2	TK 引脚
0x3	保留

- **CKO: 发送时钟输出模式选择**

CKO	发送时钟输出模式	TK 引脚
0x0	无	单输入
0x1	连续发送时钟	输出
0x2-0x7	保留	

- **CKI: 发送时钟反转**

0：发送时钟下降沿数据与帧同步信号移出。

1：发送时钟上升沿数据与帧同步信号移出。

CKI 只影响发送时钟，对输出时钟信号无影响。

- **START: 发送启动选择**

START	发送启动
0x0	连续，一旦将字写入 SSC_THR 寄存器 (若发送使能)，在前一个数据发送结束后立即启动
0x1	接收启动
0x2	检测 TF 低电平信号
0x3	检测 TF 高电平信号
0x4	检测 TF 下降沿信号
0x5	检测 TF 上升沿信号
0x6	检测 TF 电平变化信号
0x7	检测 TF 边沿信号
0x8-0xF	保留

- **STTDLY: 发送启动延迟**

- **STTDLY: 发送启动延迟**

若 STTDLY 不为 0，在启动事件与实际开始接收间插入一个 STTDLY 时钟周期的延迟。当发送器编程与接收器同步时，延迟依然有效。

请注意：设置 STTDLY 时需非常小心。若 STTDLY 与 TAG (发送同步数据) 相比太短，数据取代 TAG 结束发射。

- **PERIOD: 发送周期分频器选择**

该域选择应用于所选时钟的分频器，以产生一个新的帧同步信号。若为 0，不产生 PERIOD 信号；若不为 0，每 2 x (PERIOD+1) 个发送时钟产生一个 PERIOD 信号。

SSC 发送帧模式寄存器

寄存器名称： SSC_TFMR

访问类型： 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	FSEDGE
23	22	21	20	19	18	17	16
FSDEN	FSOS			FSLEN			
15	14	13	12	11	10	9	8
–	–	–	–	DATNB			
7	6	5	4	3	2	1	0
MSBF	–	DATDEF	DATLEN				

- **DATLEN: 数据长度**

不支持 0x0。 DATLEN 值可设定在 0x1 到 0x1F 间。

位串包含 DATLEN + 1 数据位。此外，它定义了接收器 PDC 执行的传输大小。

若 DATLEN 小于或等于 7，数据以字节传输；若 DATLEN 在 8 到 15 间 (包括 15)，数据以半字传输；其它值时，数据以 32 位字传输。

- **DATDEF: 数据默认值**

该位定义发送输出时 TD 引脚电平。注意若引脚由 PIO 控制器定义为多驱动模式，该引脚只有在 SCC TD 输出为 1 时使能。

- **MSBF: 高位在先**

0：首先是数据寄存器中的低位移出。

1：首先是数据寄存器中的高位移出。

- **DATNB: 每帧数据数**

该域定义每次发送开始后发送的数据字数。若为 0，只传输 1 个数据字，最多可传输 16 个数据字。

- **FSLEN: 发送帧同步长度**

若 FSDEN 为 1 ,该域定义发送帧同步信号长度及发送同步数据寄存器中移出的数据位数；若为 0 ,在一个发送时钟周期中发送帧同步信号，并且最多可达 16 个时钟周期脉冲长度

- **FSOS: 发送帧同步输出选择**

FSOS	选定的发送帧同步信号	TF 引脚
0x0	无	单输入
0x1	负脉冲	输出
0x2	正脉冲	输出
0x3	数据传输时拉低	输出
0x4	数据传输时拉高	输出
0x5	数据传输时切换	输出
0x6-0x7	保留	未定义

- **FSDEN: 帧同步数据使能**

0：发送帧同步信号时 TD 线驱动到默认值。

1：发送帧同步信号传输时移出 SSC_TSHR 值。

- **FSEDGE: 帧同步边沿检测**

确定帧同步边沿，在 SSC 状态寄存器的 TXSYN 中设置。

FSEDGE	帧同步边沿检测
0x0	正沿检测
0x1	负沿检测



SC 接收保持寄存器

寄存器名称： SSC_RHR

访问类型： 只读

31	30	29	28	27	26	25	24
RDAT							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
RDAT							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RDAT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RDAT							

- **RDAT: 接收数据**

不管 SSC_RFMR 寄存器中 DATLEN 定义的数据位数为多少均右对齐。

SSC 发送保持寄存器

寄存器名称： SSC_THR

访问类型： 只写

31	30	29	28	27	26	25	24
TDAT							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
TDAT							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
TDAT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TDAT							

TDAT: 发送数据

不管 SSC_TFMR 寄存器中 DATLEN 定义的数据位数为多少均右对齐。

SSC 接收同步保持寄存器

寄存器名称： SSC_RSHR

访问类型： 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RSDAT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RSDAT							

- **RSDAT: 接收同步数据**

不管 SSC_RFMR 寄存器中 FSLEN 定义的数据位数为多少均右对齐。

SSC 发送同步保持寄存器

寄存器名称： SSC_TSHR

访问类型： 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
TSDAT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TSDAT							

- **TSDAT: 发送同步数据**

不管 SSC_TFMR 寄存器中 FSLEN 定义的数据位数为多少均右对齐。

SSC 状态寄存器

寄存器名称 : SSC_SR

访问类型 : 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	RXEN	TXEN
15	14	13	12	11	10	9	8
–	–	–	–	RXSYN	TXSYN	–	–
7	6	5	4	3	2	1	0
RXBUFF	ENDRX	OVRUN	RXRDY	TXBUFE	ENDTX	TXEMPTY	TXRDY

- **TXRDY: 发送就绪**

0 : 数据载入 SSC_THR 并等待载入发送移位寄存器。

1 : SSC_THR 为空。

- **TXEMPTY: 发送空**

0 : 数据仍在 SSC_THR 中或正从发送移位寄存器发送。

1 : 写入 SSC_THR 的最后一个数据已载入发送移位寄存器并被其送出。

- **ENDTX: 发送结束**

0 : 上次写 SSC_TCR 或 SSC_TNCR 后寄存器 SSC_TCR 未达到 0。

1 : 上次写 SSC_TCR 或 SSC_TNCR 后寄存器 SSC_TCR 已达到 0。

- **TXBUFE: 发送缓冲器空**

0 : SSC_TCR 或 SSC_TNCR 值不为 0。

1 : SSC_TCR 与 SSC_TNCR 值均为 0。

- **RXRDY: 接收就绪**

0 : SSC_RHR 为空。

1 : 数据已被接收并载入 SSC_RHR。

- **OVRUN: 接收溢出**

0 : 上次读状态寄存器后，前一数据未被读取时无数据载入 SSC_RHR。

1 : 上次读状态寄存器后，前一数据未被读取时已有数据载入 SSC_RHR。

- **ENDRX: 接收结束**

0 : 数据写入接收计数寄存器或接收下一计数寄存器。

1 : 当接收计数寄存器达到 0， PDC 传输结束。

- **RXBUFF: 接收缓冲器满**

0 : SSC_RCR 或 SSC_RNCR 值不为 0。

1 : SSC_RCR 与 SSC_RNCR 值均为 0。

- **TXSYN: 发送同步**

0 : 上次读状态寄存器后未出现 Tx 同步。

1 : 上次读状态寄存器后出现 Tx 同步。

- **RXSYN: 接收同步**

0 : 上次读状态寄存器后未出现 Rx 同步。

1 : 上次读状态寄存器后出现 Rx 同步。

- **TXEN: 发送使能**

0：发送数据禁用。

1：发送数据使能。

- **RXEN: 接收使能**

0：接收数据禁用。

1：接收数据使能。



SSC 中断使能寄存器

寄存器名称 :SSC_IER

访问类型 : 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	RXSYN	TXSYN	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RXBUFF	ENDRX	OVRUN	RXRDY	TXBUFE	ENDTX	TXEMPTY	TXRDY

- **TXRDY: 发送就绪**
- **TXEMPTY: 发送空**
- **ENDTX: 发送结束**
- **TXBUFE: 发送缓冲器空**
- **RXRDY: 接收就绪**
- **OVRUN: 接收移出**
- **ENDRX: 接收结束**
- **RXBUFF: 接收缓冲器满**
- **TXSYN: Tx 同步**
- **RXSYN: Rx 同步**

0 : 无效。

1 : 使能相应中断。

SSC 中断禁用寄存器

寄存器名称： SSC_IDR

访问类型： 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	RXSYN	TXSYN	–	–
7	6	5	4	3	2	1	0
RXBUFF	ENDRX	OVRUN	RXRDY	TXBUFE	ENDTX	TXEMPTY	TXRDY

- **TXRDY:** 发送就绪
- **TXEMPTY:** 发送空
- **ENDTX:** 发送结束
- **TXBUFE:** 发送缓冲器空
- **RXRDY:** 接收就绪
- **OVRUN:** 接收移出
- **ENDRX:** 接收结束
- **RXBUFF:** 接收缓冲器满
- **TXSYN:** Tx 同步
- **RXSYN:** Rx 同步

0：无效。

1：禁用相应中断。



SSC 中断屏蔽寄存器

寄存器名称： SSC_IMR

访问类型： 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	RXSYN	TXSYN	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RXBUFF	ENDRX	OVRUN	RXRDY	TXBUFE	ENDTX	TXEMPTY	TXRDY

- **TXRDY: 发送就绪**
- **TXEMPTY: 发送空**
- **ENDTX: 发送结束**
- **TXBUFE: 发送缓冲器空**
- **RXRDY: 接收就绪**
- **OVRUN: 接收移出**
- **ENDRX: 接收结束**
- **RXBUFF: 接收缓冲器满**
- **TXSYN: Tx 同步**
- **RXSYN: Rx 同步**

0：相应中断禁用。

1：相应中断使能。

定时器 / 计数器 (TC)

概述

定时器 / 计数器 (TC) 包括三个 相同的 16 位定时器 / 计数器通道。

每个通道可独立编程，以完成不同功能，包括 : 频率测量、事件计数、间隔测量、脉冲产生、延迟时间及脉宽调制。

每个通道有三个外部时钟输入，五个内部时钟输入及两个可由用户配置的多功能输入 / 输出信号。每个通道驱动一个可编程内部中断信号来产生处理器中断。

定时器 / 计数器有两个作用于这三个 TC 通道的全局寄存器。

块控制寄存器允许使用同样的指令同时启动三个通道。

块模式寄存器为每个通道定义外部时钟输入，允许将它们链接。

定时器 / 计数器主要特性如下：

- 三个 16 位定时器 / 计数器通道
- 多种功能，包括：
  - 频率测量
  - 事件计数
  - 间隔测量
  - 脉冲产生
  - 延迟时间
  - 脉宽调制
  - 向上 / 向下能力
- 每个通道均为用户可配置，包括：
  - 三个外部时钟输入
  - 五个内部时钟输入
  - 两个多功能输入 / 输出信号
- 内部中断信号

两个全局寄存器可在所有三个 TC 通道中使用



方框图

Figure 220. 定时器 / 计数器框图

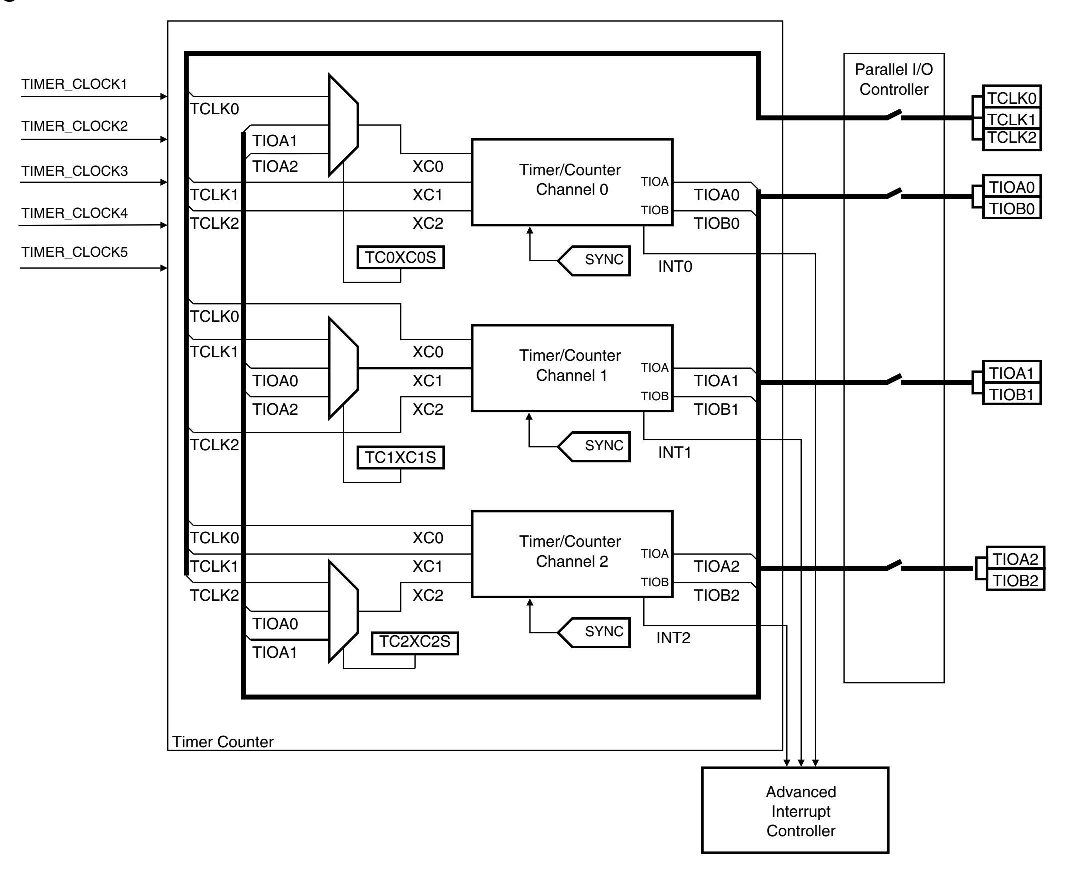


Table 89. 信号名称说明

块 / 通道	信号名称	说明
通道信号	XC0, XC1, XC2	外部时钟输入
	TIOA	捕获模式：定时器 / 计数器输入 波形模式：定时器 / 计数器输出
	TIOB	捕获模式：定时器 / 计数器输入 波形模式：定时器 / 计数器输入 / 输出
	INT	中断信号输出
	SYNC	同步输入信号
块信号	TCLK0, TCLK1, TCLK2	外部时钟输入
	TIOA0	通道 0 的 TIOA 信号
	TIOB0	通道 0 的 TIOB 信号
	TIOA1	通道 1 的 TIOA 信号
	TIOB1	通道 1 的 TIOB 信号
	TIOA2	通道 2 的 TIOA 信号
	TIOB2	通道 2 的 TIOB 信号

引脚名称列表

Table 90. TC 引脚列表

引脚名称	说明	类型
TCLK0-TCLK2	外部时钟输入	输入
TIOA0-TIOA2	I/O 线 A	I/O
TIOB0-TIOB2	I/O 线 B	I/O

附属产品

关于 TC 硬件细节，见相关产品手册。

I/O 线

连接外设的引脚可与 PIO 线复用。必须先对 PIO 控制器编程，给 TC 引脚分配外设功能。

电源管理

TC 由电源管理控制器 (PMC) 定时，因此必须先配置 PMC 以使能定时器 / 计数器时钟。

中断

TC 有一条与高级中断控制器 (AIC) 连接的中断线。处理中断前需要先对 AIC 编程，然后再配置 TC。

功能说明

TC 说明

TC 的三个通道相互独立，但操作相同。通道寄存器编程见 Table 90 on page 467。

16 位计数器

每通道有一个 16 位寄存器。寄存器值在所选时钟每个正沿处自减。当计数器达到 0xFFFF 并转为 0x0000 时，表明发生溢出， TC_SR (状态寄存器) 中 COVFS 位置位。

计数器当前值可通过计数器值寄存器 TC_CV 实时读取。计数器由触发器复位。此时，计数器值在下一个选定时钟有效边沿时转为 0x0000。

时钟选择

在块级上，每个通道输入时钟通过对 TC_BMR (块模式) 编程可与外部输入 TCLK0、 TCLK1、TCLK2 或可配置的 I/O 信号 TIOA0、 TIOA1、 TIOA2 连接，见 Figure 221。

每个通道可独立选择内部或外部计数器时钟源：

- 内部时钟源：TIMER_CLOCK1、 TIMER_CLOCK2、 TIMER_CLOCK3、 TIMER_CLOCK4、 TIMER_CLOCK5
- 外部时钟信号：XC0、 XC1 或 XC2

选择由 TC 通道模式寄存器的 TCCLKS 位完成。

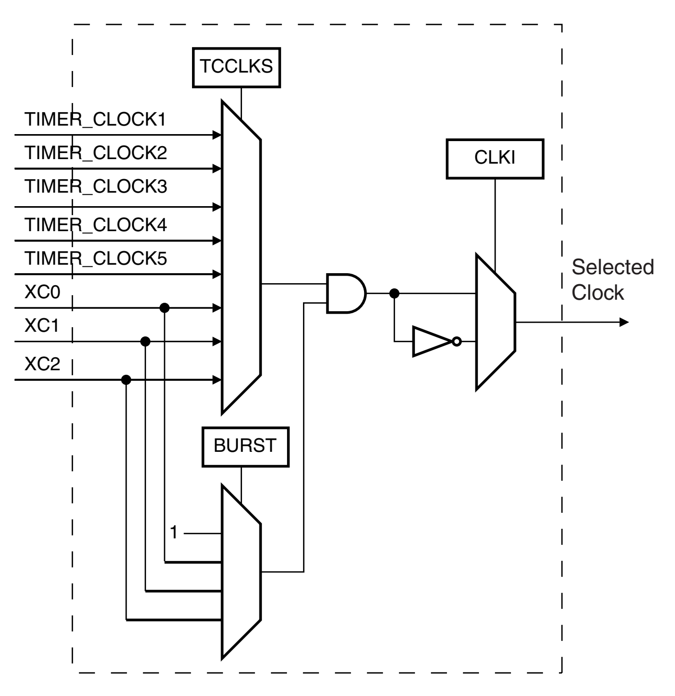
所选时钟可通过 TC_CMR 寄存器的 CLKI 位实现反转。因此可使用时钟负沿进行计数。

脉冲功能使外部信号为高时时钟有效。模式寄存器中的 BURST 参数定义该信号 (无、 XC0、XC1、 XC2)。

Note: 使用外部时钟时，每个电平持续时间必须比主机时钟周期长。主机时钟频率至少为外部时钟频率的 2.5 倍。



Figure 221. 时钟选择

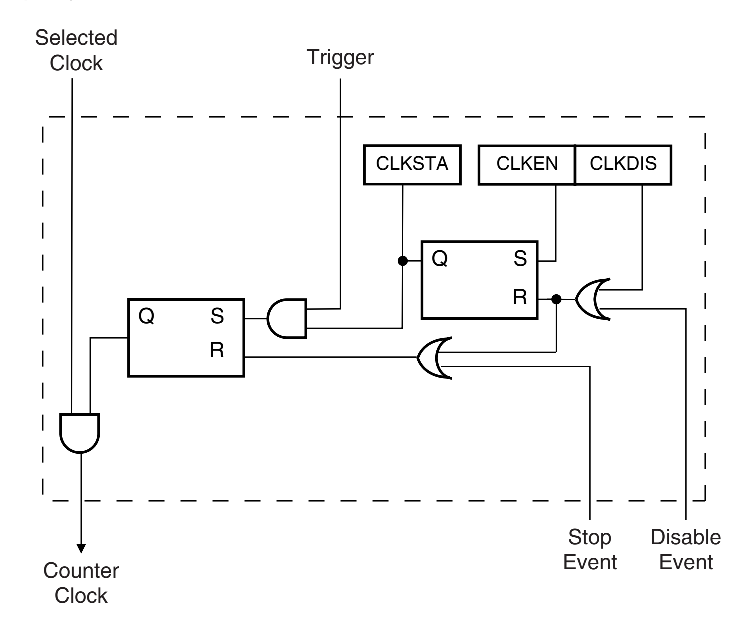


时钟控制

每个计数器时钟有两种控制方式：可使能 / 禁用或启动 / 停止，见 Figure 222。

- 用户可使用控制寄存器的 CLKEN 与 CLKDIS 命令使能或禁用时钟。捕获模式下，若 TC_CMR 中 LDBDIS 位置为 1 , 通过 RB 载入事件将时钟禁用。波形模式下 , 若 TC_CMR 中 CPCDIS 位置为 1，通过 RC 比较事件将时钟禁用。当时钟禁用时，启动与停止命令无效，只有控制寄存器的 CLKEN 命令可将时钟重新使能。当时钟使能后，置位状态寄存器的 CLKSTA 位。
- 时钟也可启动或停止：触发器 (软件、同步、外部或比较) 用来启动时钟。捕获模式下，时钟由 RB 载入事件停止 (TC_CMR 中 LDBSTOP = 1) ；波形模式下，时钟由 RC 比较事件停止 (TC_CMR 中 CPCSTOP = 1)。只有当时钟使能时启动与停止命令才有效。

Figure 222. 时钟控制



TC 操作模式

每个通道可工作在两种不同模式下：

- 捕获模式提供信号测量。
- 波形模式用来产生波形。

TC 操作模式由 TC 通道模式寄存器的 WAVE 位编程设定。

捕获模式下， TIOA 与 TIOB 配置为输入。

波形模式下， TIOA 配置为输出，若未选择外部触发器 TIOB 也为输出。

触发器

触发器复位计数器并启动计数器时钟。两种模式下有三种通用触发器，第四种外部触发器分别适用于每种模式。

下列触发器适用于两种模式：

- 软件触发器：每个通道有一软件触发器，通过设置 TC_CCR 中的 SWTRG 有效。
- SYNC：每个通道有一个同步信号 SYNC。当出现时，该信号与软件触发器效果相同。通过写 TC_BCR(块控制)，所有通道的 SYNC 信号同时出现。
- 比较 RC 触发器：RC 在每个通道中执行，若 TC_CMR 中 CPCTRG 置位，能在计数器值等于 RC 值时提供触发。

通道也能配置为有一个外部触发器。捕获模式下，在 TIOA 与 TIOB 信号间选择外部触发信号；波形模式下，外部事件可在下列信号上编程：TIOB、 XC0、 XC1 或 XC2。通过设置 TC_CMR 中的 ENETRG 可实现外部事件执行触发。

若使用外部触发器，脉冲持续时间必须比主机时钟周期长，以便对其检测。

不管触发器是否使用，它将在后续选定时钟的有效沿记录。即触发后计数器值即不为零，特别是当选定低频信号作为时钟时。

捕获工作模式

清除 TC_CMR (通道模式寄存器) 的 WAVE 参数即可进入该模式。

捕获模式允许TC通道对脉冲时间、频率、周期占空比及作为输入的TIOA与TIOB信号进行测量。

Figure 223 给出捕获模式下 TC 通道配置。

捕获寄存器 A 与 B

寄存器 A 与 B (RA 与 RB) 作为捕获寄存器使用。即当可编程事件在 TIOA 上出现时，它们将载入计数器值。

TC_CMR中的LDRA参数定义寄存器A的载入TIOA边沿；LDRB参数定义寄存器B的载入TIOA边沿。

只有在最近触发后未载入或 RB 已载入时， RA 才会载入。

RB 只有在 RA 已载入或最后载入 RB 时，才会载入。

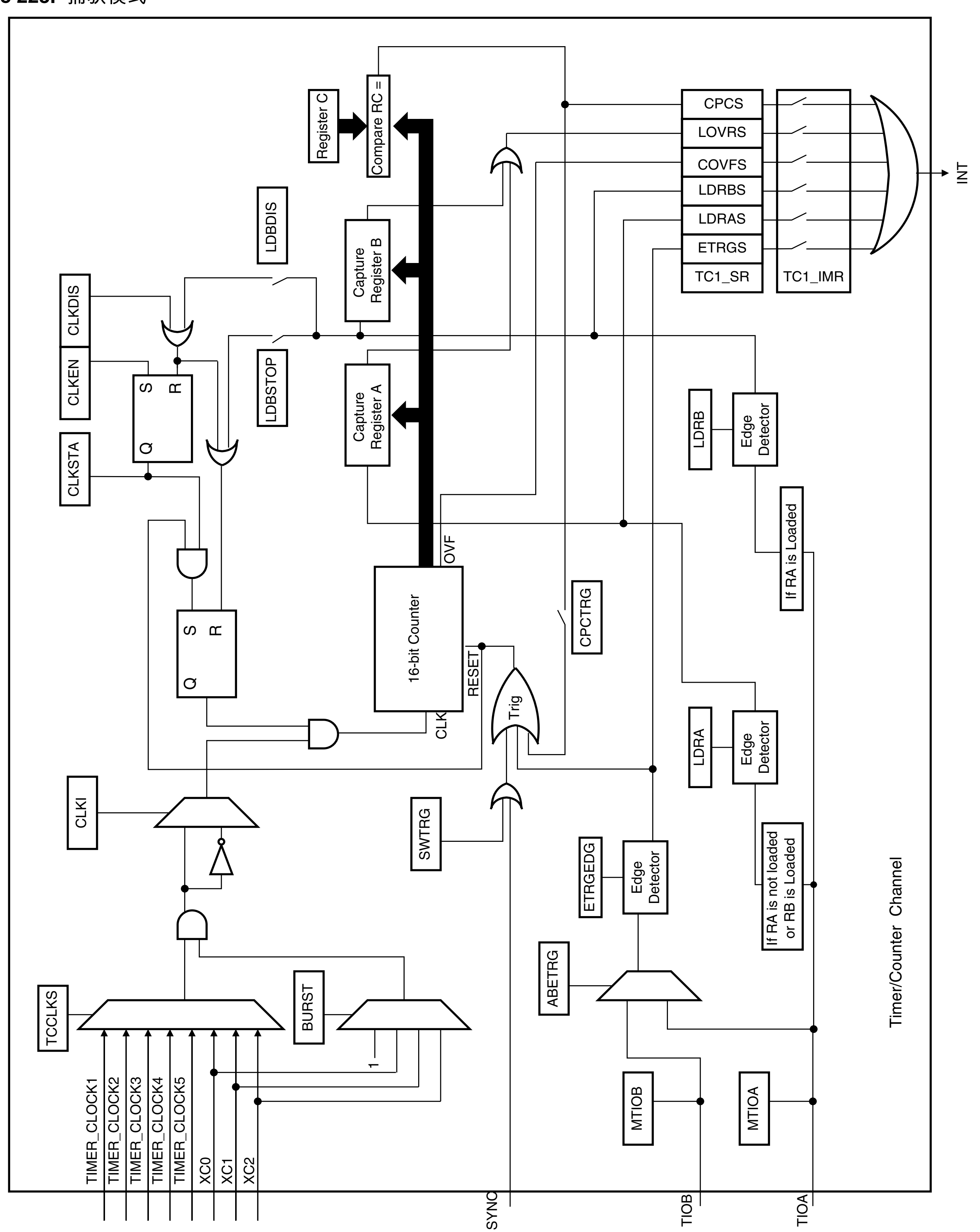
读最近载入值前载入 RA 或 RB 将设置 TC_SR (状态寄存器) 的溢出错误标志 (LOVRS)。此时将旧值覆盖。

触发条件

除 SYNC 信号、软件触发器及 RC 比较寄存器外，还可定义外部触发。

TC_CMR寄存器的ABETRG位选择使用TIOA或TIOB输入信号作为外部触发。ETRGEDG参数定义产生外部触发的检测边沿(上升沿、下降沿或两者皆可)。若ETRGEDG = 0 ,外部触发禁用。

Figure 223. 捕获模式



波形工作模式

通过设置 TC_CMR 寄存器的 WAVE 参数进入波形工作模式。

波形工作模式下， TC 通道产生 1 个或 2 个相同频率的可独立编程占空比的 PWM 信号，或产生不同类型的单发射或重复脉冲。

该模式下，TIOA 配置为输出， TIOB 在未使用外部事件时也定义为输出 (TC_CMR 中的 EEVT 参数)。

Figure 224 给出波形工作模式下的 TC 通道配置。

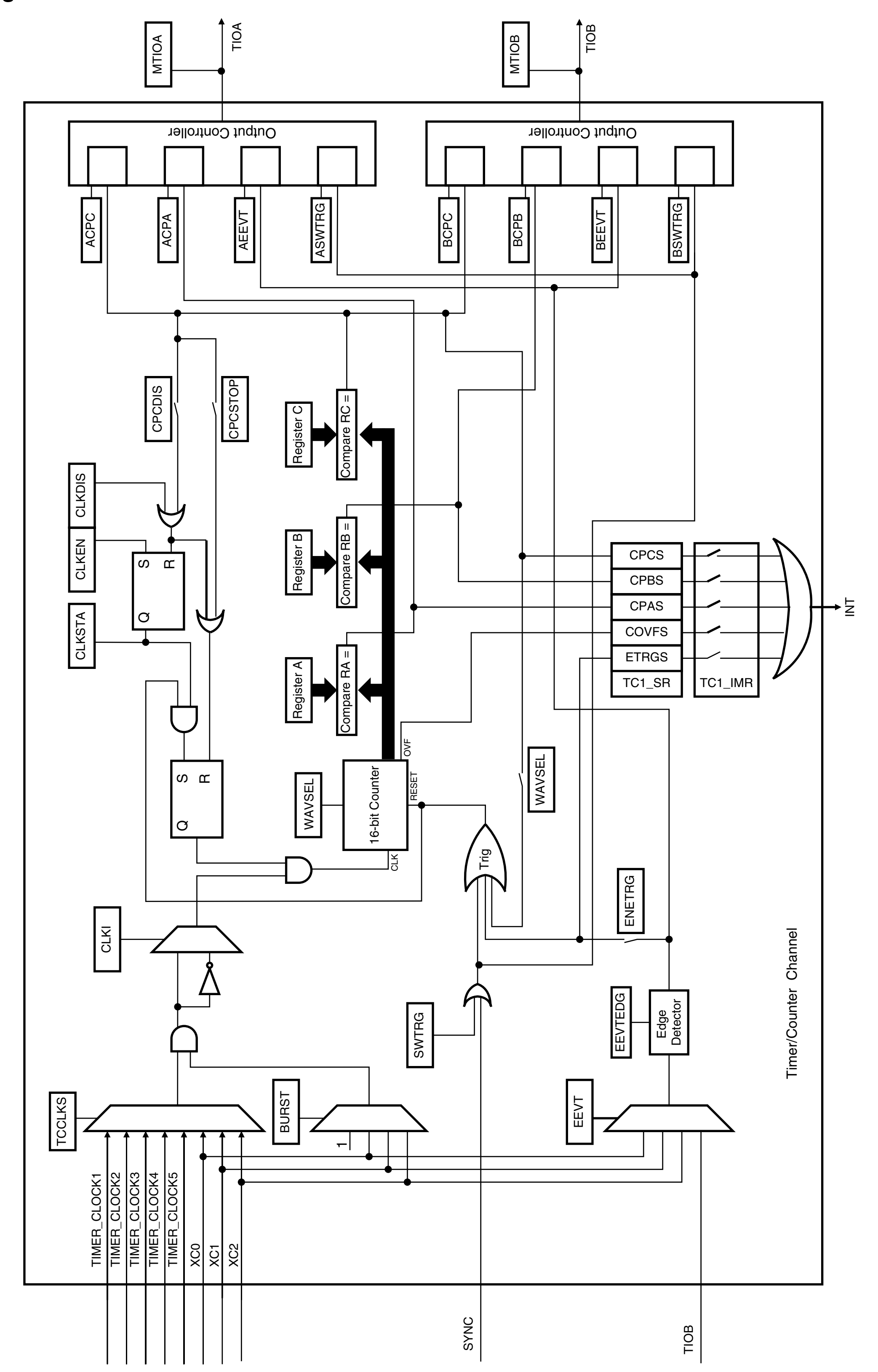
波形选择

根据 TC_CMR 中 WAVSEL 参数的不同，TC_CV 动作不同。

任何情况下， RA、 RB 及 RC 可作为比较寄存器使用。

RA比较器用来控制TIOA输出 ,RB比较用来控制TIOB输出(若配置正确) ,RC比较用来控制TIOA 与 / 或 TIOB 输出。

Figure 224. 波形模式





WAVSEL = 00

当 WAVSEL = 00 时， TC_CV 值由 0 增加到 0xFFFF。一旦达到 0xFFFF， TC_CV 值复位。TC_CV 值重新增加且继续循环，见 Figure 225。

外部事件触发或软件触发可复位 TC_CV 值。注意，触发可能随时出现，见 Figure 226。

该配置下 RC 比较不能编程产生触发。同时 RC 比较可停止计数器时钟 (TC_CMR 中 CPCSTOP = 1) 与 / 或禁用计数器时钟 (TC_CMR 中 CPCDIS = 1)。

Figure 225. WAVSEL= 00 无触发

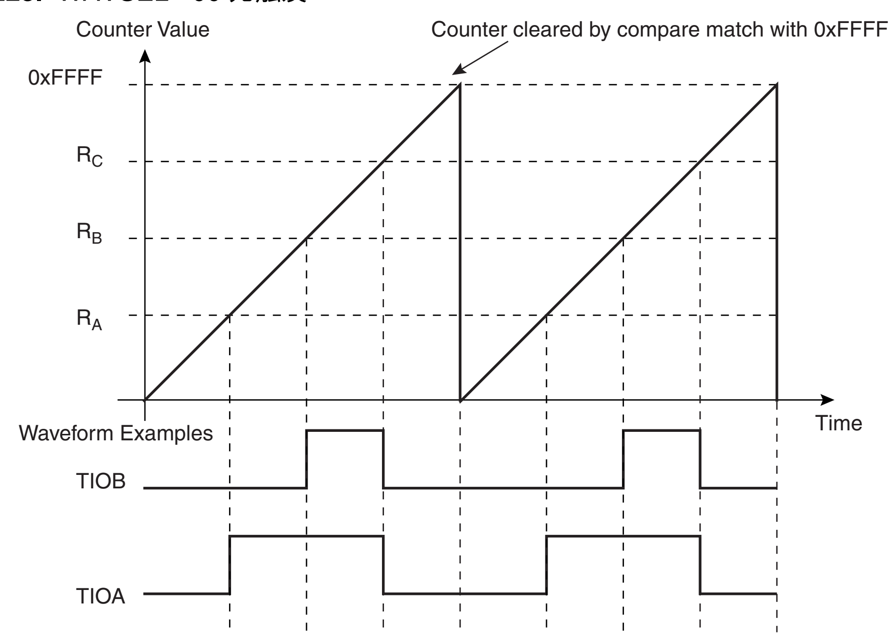
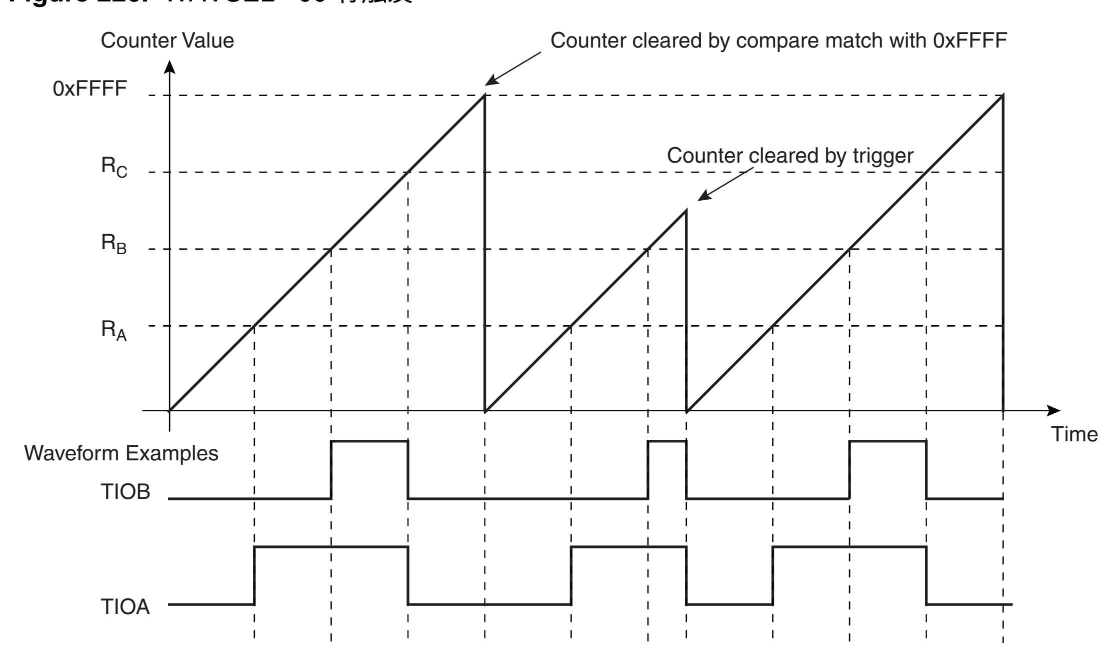


Figure 226. WAVSEL= 00 有触发



WAVSEL = 10

当 WAVSEL = 10 时， TC_CV 值由 0 增加到 RC 值，然后自动复位。一旦 TC_CV 值复位，它开始重新循环，见 Figure 227。

若外部事件或软件触发器编程正确， TC_CV 可随时复位，见 Figure 228。

此外，RC 比较可停止计数器时钟 (TC_CMR 中 CPCSTOP = 1) 与 / 或禁用计数器时钟 (TC_CMR 中 CPCDIS = 1)。


Figure 227. WAVSEL = 10 无触发

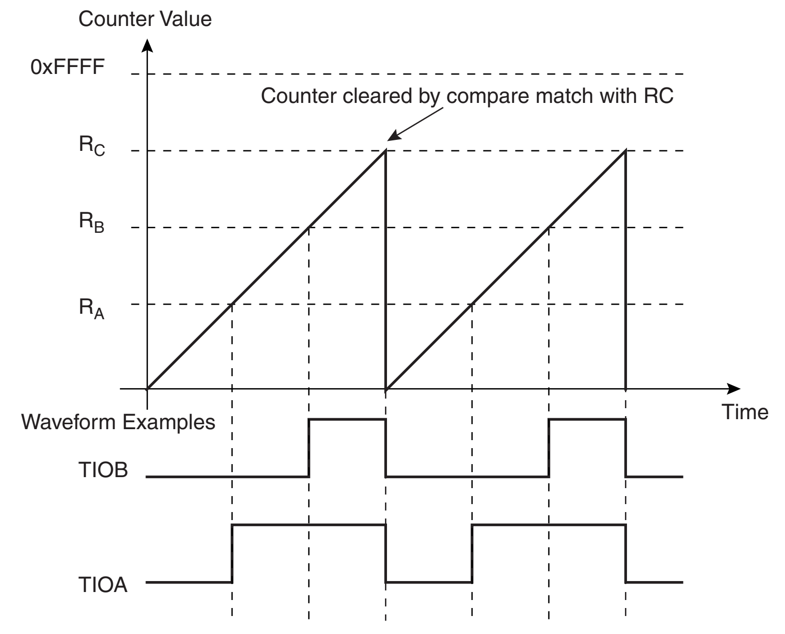
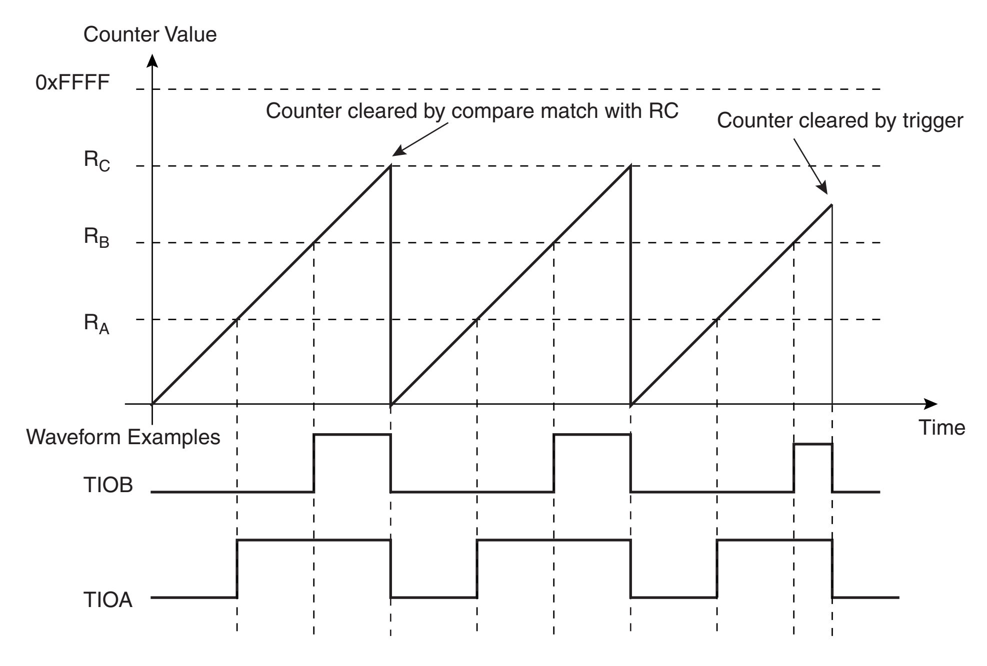


Figure 228. WAVSEL = 10 有触发



WAVSEL = 01

当 WAVSEL = 01， TC_CV 值由 0 增加到 0xFFFF。一旦达到 0xFFFF， TC_CV 值减到 0，再重新增加到 0xFFFF，如此循环下去，见 Figure 229。

外部事件或软件触发可随时修改 TC_CV。若 TC_CV 增加时触发出现， TC_CV 开始递减；若 TC_CV 递减时收到触发， TC_CV 开始递增，见 Figure 230。

该配置下，不可对 RC 比较编程产生触发。

RC 比较可停止计数器时钟 (CPCSTOP = 1) 与 / 或禁用计数器时钟 (CPCDIS = 1)。

Figure 229. WAVSEL = 01 无触发

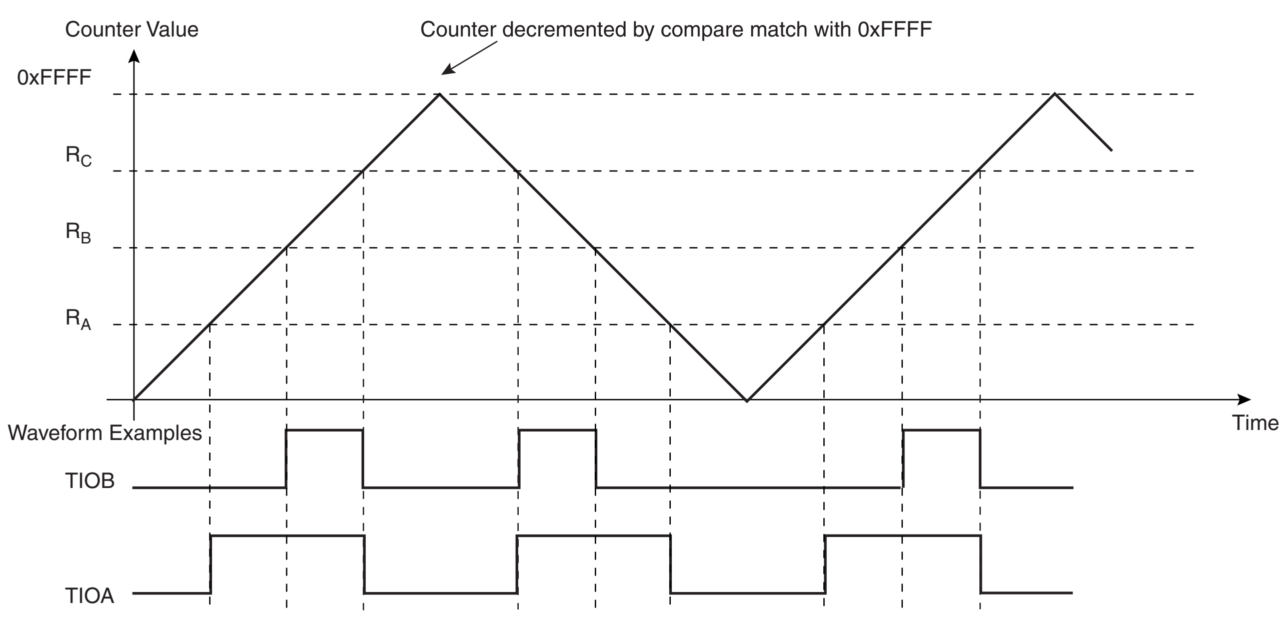
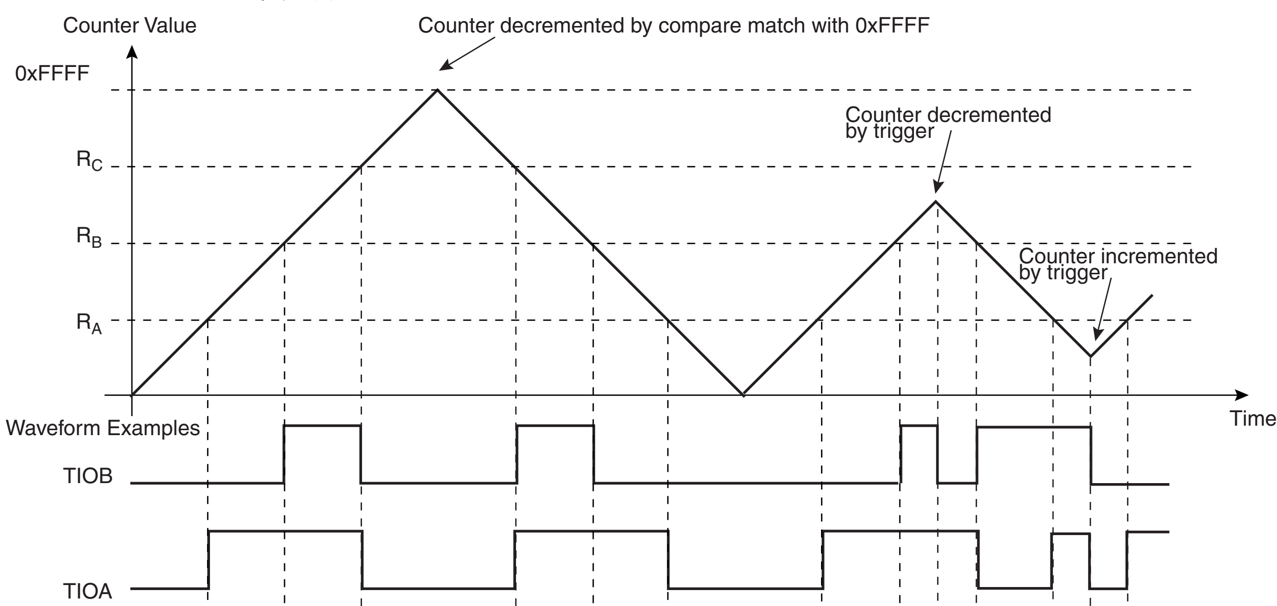


Figure 230. WAVSEL = 01 有触发



WAVSEL = 11

当 WAVSEL = 11， TC_CV 值由 0 增加到 RC。一旦达到 RC， TC_CV 值递减到 0，然后重新递增到 RC，如此循环下去，见 Figure 231。

外部事件或软件触发可随时修改 TC_CV。若 TC_CV 增加时触发出现， TC_CV 开始递减；若 TC_CV 递减时收到触发， TC_CV 开始递增，见 Figure 232。

RC 比较可停止计数器时钟 (CPCSTOP = 1) 与 / 或禁用计数器时钟 (CPCDIS = 1)。

Figure 231. WAVSEL = 11 无触发

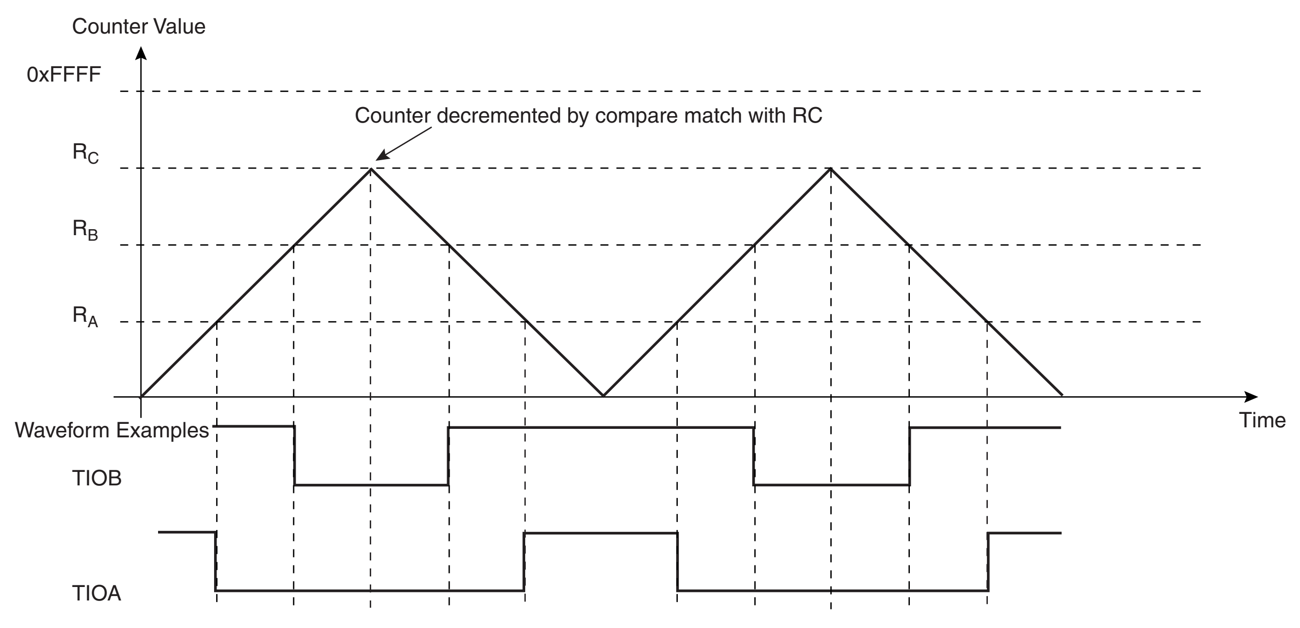
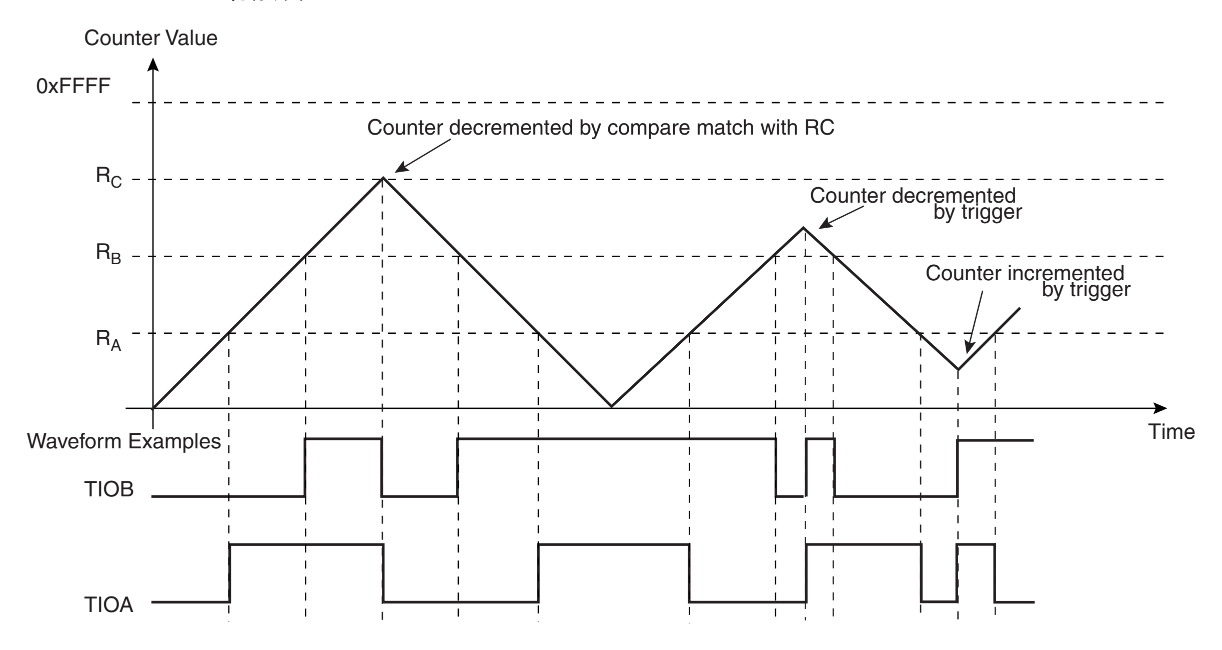


Figure 232. WAVSEL = 11 有触发



外部事件 / 触发条件

可对外部事件编程来检测时钟源 (XC0、XC1、XC2) 或 TIOB。然后可选择外部事件作为触发。

TC_CMR 中 EEVT 参数用来选择外部触发器。参数 EEVTEDG 定义每个可能的外部触发边沿 (上升沿、下降沿或二者皆可)。若 EEVTEDG 清零，不定义外部事件。

若定义 TIOB 为外部事件信号 (EEVT = 0) ,TIOB 不再作为输出且 TC 通道只可产生 TIOA 波形。

若定义了外部事件，通过设置 TC_CMR 中的 ENETRG 位，外部事件可作为触发器。

与捕获模式相同， SYNC 信号与软件触发作为触发器同样有效。若 WAVSEL 参数设置恰当，RC 比较也可作为触发器。



输出控制器

输出控制器定义事件后 TIOA 与 TIOB 输出电平变化。只有当 TIOB 定义为输出时使用 TIOB 控制 (不是外部事件)。

下列事件控制 TIOA 与 TIOB：软件触发器、外部事件及 RC 比较。 RA 比较控制 TIOA ， RB 比较控制 TIOB。根据相应的 TC_CMR 定义，每个事件可编程设置、清除或切换输出。

定时器 / 计数器 (TC) 用户接口

Table 91. 定时器 / 计数器 (TC) 全局存储器映射

偏移	通道 / 寄存器	名称	访问类型	复位值
0x00	TC 通道 0	见 Table 92		
0x40	TC 通道 1	见 Table 92		
0x80	TC 通道 2	见 Table 92		
0xC0	TC 块控制寄存器	TC_BCR	只写	–
0xC4	TC 块模式寄存器	TC_BMR	读 / 写	0

TC_BCR (块控制寄存器) 与 TC_BMR (块控制寄存器) 控制整个 TC 块。TC 通道由 Table 92 列出的寄存器控制。Table 92 中每个通道寄存器偏移与 Table 92 中提到的相应通道偏移相关。

Table 92. 定时器 / 计数器 (TC) 通道存储器映射

偏移	寄存器	名称	访问类型	复位值
0x00	通道控制寄存器	TC_CCR	只写	–
0x04	通道模式寄存器	TC_CMR	读 / 写	0
0x08	保留		–	–
0x0C	保留		–	–
0x10	计数器值	TC_CV	只读	0
0x14	寄存器 A	TC_RA	读 / 写 (1)	0
0x18	寄存器 B	TC_RB	读 / 写 (1)	0
0x1C	寄存器 C	TC_RC	读 / 写	0
0x20	状态寄存器	TC_SR	只读	0
0x24	中断使能寄存器	TC_IER	只写	–
0x28	中断禁用寄存器	TC_IDR	只写	–
0x2C	中断屏蔽寄存器	TC_IMR	只读	0

Notes: 1. WAVE = 0 时读。



TC 块控制寄存器

寄存器名称： TC_BCR

访问类型： 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	–	–	–	–	–	SYNC

- **SYNC: 同步命令**

0 = 无效。

1 = 出现 SYNC 信号，给每个通道同时产生软件触发。

TC 块模式寄存器

寄存器名称 : TC_BMR

访问类型 : 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	TC2XC2S		TCXC1S		TC0XC0S	

- **TC0XC0S: 外部时钟信号 0 选择**

TC0XC0S		信号与 XC0 连接
0	0	TCLK0
0	1	无
1	0	TIOA1
1	1	TIOA2

- **TC1XC1S: 外部时钟信号 1 选择**

TC1XC1S		信号与 XC1 连接
0	0	TCLK1
0	1	无
1	0	TIOA0
1	1	TIOA2

- **TC2XC2S: 外部时钟信号 2 选择**

TC2XC2S		信号与 XC2 连接
0	0	TCLK2
0	1	无
1	0	TIOA0
1	1	TIOA1



TC 通道控制寄存器

寄存器名称 : TC_CCR

访问类型 : 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	SWTRG	CLKDIS	CLKEN

- **CLKEN: 计数器时钟使能命令**

0 = 无效。

1 = 若 CLKDIS 不为 1，使能时钟。

- **CLKDIS: 计数器时钟禁用命令**

0 = 无效。

1 = 禁用时钟。

- **SWTRG: 软件触发命令**

0 = 无效。

1 = 软件触发执行：计数器复位，时钟启动。

TC 通道模式寄存器：捕获模式

寄存器名称： TC_CMR

访问类型： 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	LDRB		LDRA	

15	14	13	12	11	10	9	8
WAVE = 0	CPCTRG	–	–	–	ABETRG	ETRGEDG	

7	6	5	4	3	2	1	0
LDBDIS	LDBSTOP	BURST		CLKI	TCCLKS		

- **TCCLKS: 时钟选择**

TCCLKS			选定时钟
0	0	0	TIMER_CLOCK1
0	0	1	TIMER_CLOCK2
0	1	0	TIMER_CLOCK3
0	1	1	TIMER_CLOCK4
1	0	0	TIMER_CLOCK5
1	0	1	XC0
1	1	0	XC1
1	1	1	XC2

- **CLKI: 时钟反转**

0 = 时钟上升沿计数器增加。

1 = 时钟下降沿计数器增加。

- **BURST: 脉冲信号选择**

BURST		
0	0	时钟不是由外部时钟开启
0	1	XC0 与选定时钟相与
1	0	XC1 与选定时钟相与
1	1	XC2 与选定时钟相与

- **LDBSTOP: RB 载入时，计数器时钟停止**

0 = 当 RB 载入出现时，计数器时钟未停止。

1 = 当 RB 载入出现时，计数器时钟停止。

- **LDBDIS: RB 载入时，计数器时钟禁用**

0 = 当 RB 载入出现时，计数器时钟未禁用。

1 = 当 RB 载入出现时，计数器时钟禁用。



- **ETRGEDG: 外部触发边沿选择**

ETRGEDG		边沿
0	0	无
0	1	上升沿
1	0	下降沿
1	1	任意沿

- **ABETRG: TIOA 或 TIOB 外部触发选择**

0 = TIOB 用作外部触发器。

1 = TIOA 用作外部触发器。

- **CPCTRG: RC 比较触发使能**

0 = RC 比较对计数器及其时钟无效。

1 = RC 比较复位计数器并启动计数器时钟。

- **WAVE**

0 = 捕获模式使能。

1 = 捕获模式禁用 (波形模式使能)。

- **LDRA: RA 载入选择**

LDRA		边沿
0	0	无
0	1	TIOA 上升沿
1	0	TIOA 下降沿
1	1	TIOA 任意沿

- **LDRB: RB 载入选择**

LDRB		边沿
0	0	无
0	1	TIOA 上升沿
1	0	TIOA 下降沿
1	1	TIOA 任意沿

TC 通道模式寄存器：波形模式

寄存器名称： TC_CMR

访问类型： 读 / 写

31	30	29	28	27	26	25	24
BSWTRG		BEEVT		BCPC		BCPB	

23	22	21	20	19	18	17	16
ASWTRG		AEEVT		ACPC		ACPA	

15	14	13	12	11	10	9	8
WAVE = 1	WAVSEL		ENETRG	EEVT		EEVTEDG	

7	6	5	4	3	2	1	0
CPCDIS	CPCSTOP	BURST		CLKI	TCCLKS		

- **TCCLKS: 时钟选择**

TCCLKS			选定时钟
0	0	0	TIMER_CLOCK1
0	0	1	TIMER_CLOCK2
0	1	0	TIMER_CLOCK3
0	1	1	TIMER_CLOCK4
1	0	0	TIMER_CLOCK5
1	0	1	XC0
1	1	0	XC1
1	1	1	XC2

- **CLKI: 时钟反转**

0 = 时钟上升沿计数器增加。

1 = 时钟下降沿计数器增加。

- **BURST: 脉冲信号选择**

BURST		
0	0	时钟不是由外部时钟开启
0	1	XC0 与选定时钟相与
1	0	XC1 与选定时钟相与
1	1	XC2 与选定时钟相与

- **CPCSTOP: RC 比较时，计数器时钟停止**

0 = 当计数器值达到 RC 时计数器时钟不停止。

1 = 当计数器值达到 RC 时计数器时钟停止。

- **CPCDIS: RC 比较时，计数器时钟禁用**

0 = 当计数器值达到 RC 时计数器时钟不禁用。

1 = 当计数器值达到 RC 时计数器时钟禁用。



- **EEVTEDG: 外部事件边沿选择**

EEVTEDG		边沿
0	0	无
0	1	上升沿
1	0	下降沿
1	1	任意沿

- **EEVT: 外部事件选择**

EEVT		信号选择外部事件	TIOB 方向
0	0	TIOB	输入 [1]
0	1	XC0	输出
1	0	XC1	输出
1	1	XC2	输出

Note: 1. *若选择 TIOB 作为外部事件信号，将其配置为输入并不再产生波形。*

- **ENETRG: 外部事件触发使能**

0 = 外部事件对计数器及其时钟无效。此时，所选外部事件仅控制 TIOA 输出。

1 = 外部事件复位计数器并启动计数器时钟。

- **WAVSEL: 波形选择**

WAVSEL		效果
0	0	UP 模式，无 RC 比较自动触发
1	0	UP 模式，有 RC 比较自动触发
0	1	UPDOWN 模式，无 RC 比较自动触发
1	1	UPDOWN 模式，有 RC 比较自动触发

- **WAVE = 1**

0 = 波形模式禁用 (捕获模式使能)。

1 = 波形模式使能。

- **ACPA: TIOA 上 RA 比较效果**

ACPA		效果
0	0	无
0	1	置位
1	0	清零
1	1	切换

- **ACPC: TIOA 上 RC 比较效果**

ACPC		效果
0	0	无
0	1	置位
1	0	清零
1	1	切换

- **AEEVT: TIOA 上外部事件效果**

AEEVT		效果
0	0	无
0	1	置位
1	0	清零
1	1	切换

- **ASWTRG: TIOA 上软件触发效果**

ASWTRG		效果
0	0	无
0	1	置位
1	0	清零
1	1	切换

- **BCPB: TIOB 上 RB 比较效果**

BCPB		效果
0	0	无
0	1	置位
1	0	清零
1	1	切换

- **BCPC: TIOB 上 RC 比较效果**

BCPC		效果
0	0	无
0	1	置位
1	0	清零
1	1	切换

- **BEEVT: TIOB 上外部事件效果**

BEEVT		效果
0	0	无
0	1	置位
1	0	清零
1	1	切换

- **BSWTRG: TIOB 上软件触发效果**

BSWTRG		效果
0	0	无
0	1	置位
1	0	清零
1	1	切换



TC 计数器值寄存器

寄存器名称： TC_CV

访问类型： 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
CV							

7	6	5	4	3	2	1	0
CV							

- **CV: 计数器值**

CV 中为实时计数器值。

TC 寄存器 A

寄存器名称： TC_RA

访问类型： 若 WAVE = 0，只读；若 WAVE = 1，读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
RA							

7	6	5	4	3	2	1	0
RA							

- **RA: 寄存器 A**

RA 中为实时寄存器 A 值。

TC 寄存器 B

寄存器名称： TC_RB

访问类型： 若 WAVE = 0，只读；若 WAVE = 1，读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RB							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RB							

- **RB: 寄存器 B**

RB 中为实时寄存器 B 值。

TC 寄存器 C

寄存器名称： TC_RC

访问类型： 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RC							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RC							

- **RC: 寄存器 C**

RC 中为实时寄存器 C 值。



TC 状态寄存器

寄存器名称 : TC_SR

访问类型 : 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	MTIOB	MTIOA	CLKSTA
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ETRGS	LDRBS	LDRAS	CPCS	CPBS	CPAS	LOVRS	COVFS

- **COVFS: 计数器溢出状态**

0 = 上次读状态寄存器后未出现计数器溢出。

1 = 上次读状态寄存器后出现计数器溢出。

- **LOVRS: 载入溢出状态**

0 = 上次读状态寄存器后未出现载入溢出，或 WAVE = 1。

1 = 若 WAVE = 0，上次读状态寄存器后，在未读取相应寄存器的情况下， RA 或 RB 已至少载入两次。

- **CPAS: RA 比较状态**

0 = 上次读状态寄存器后未出现 RA 比较，或 WAVE = 0。

1 = 若 WAVE = 1，上次读状态寄存器后出现 RA 比较。

- **CPBS: RB 比较状态**

0 = 上次读状态寄存器后未出现 RB 比较，或 WAVE = 0。

1 = 若 WAVE = 1，上次读状态寄存器后出现 RB 比较。

- **CPCS: RC 比较状态**

0 = 上次读状态寄存器后未出现 RC 比较。

1 = 上次读状态寄存器后出现 RC 比较。

- **LDRAS: RA 载入状态**

0 = 上次读状态寄存器后未出现 RA 载入，或 WAVE = 1。

1 = 若 WAVE = 0，上次读状态寄存器后出现 RA 载入。

- **LDRBS: RB 载入状态**

0 = 上次读状态寄存器后未出现 RB 载入，或 WAVE = 1。

1 = 若 WAVE = 0，上次读状态寄存器后出现 RB 载入。

- **ETRGS: 外部触发状态**

0 = 上次读状态寄存器后未出现外部触发。

1 = 上次读状态寄存器后出现外部触发。

- **CLKSTA: 时钟使能状态**

0 = 时钟禁用。

1 = 时钟使能。

- **MTIOA: TIOA 镜像**

0 = TIOA 为低。若 WAVE = 0，表示 TIOA 为低；若 WAVE = 1，表示将 TIOA 拉低。

1 = TIOA 为高，若 WAVE = 0，表示 TIOA 为高；若 WAVE = 1，表示将 TIOA 拉高。

- **MTIOB: TIOB 镜像**

0 = TIOB 为低。若 WAVE = 0，表示 TIOB 为低；若 WAVE = 1，表示将 TIOB 拉低。

1 = TIOB 为高，若 WAVE = 0，表示 TIOB 为高；若 WAVE = 1，表示将 TIOB 拉高。

TC 中断使能寄存器

寄存器名称 : TC_IER

访问类型 : 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ETRGS	LDRBS	LDRAS	CPCS	CPBS	CPAS	LOVRS	COVFS

- **COVFS: 计数器溢出**

0 = 无效。

1 = 使能计数器溢出中断。

- **LOVRS: 载入溢出**

0 = 无效。

1 = 使能载入溢出中断。

- **CPAS: RA 比较**

0 = 无效。

1 = 使能 RA 比较中断。

- **CPBS: RB 比较**

0 = 无效。

1 = 使能 RB 比较中断。

- **CPCS: RC 比较**

0 = 无效。

1 = 使能 RC 比较中断。

- **LDRAS: RA 载入**

0 = 无效。

1 = 使能 RA 载入中断。

- **LDRBS: RB 载入**

0 = 无效。

1 = 使能 RB 载入中断。

- **ETRGS: 外部触发**

0 = 无效。

1 = 使能外部触发中断。

TC 中断禁用寄存器

寄存器名称 : TC_IDR

访问类型 : 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
ETRGS	LDRBS	LDRAS	CPCS	CPBS	CPAS	LOVRS	COVFS

- **COVFS: 计数器溢出**

0 = 无效。

1 = 禁用计数器溢出中断。

- **LOVRS: 载入溢出**

0 = 无效。

1 = 禁用载入溢出中断 (若 WAVE = 0)。

- **CPAS: RA 比较**

0 = 无效。

1 = 禁用 RA 比较中断 (若 WAVE = 1)。

- **CPBS: RB 比较**

0 = 无效。

1 = 禁用 RB 比较中断 (若 WAVE = 1)。

- **CPCS: RC 比较**

0 = 无效。

1 = 禁用 RC 比较中断。

- **LDRAS: RA 载入**

0 = 无效。

1 = 禁用 RA 载入中断 (若 WAVE = 0)。

- **LDRBS: RB 载入**

0 = 无效。

1 = 禁用 RB 载入中断 (若 WAVE = 0)。

- **ETRGS: 外部触发**

0 = 无效。

1 = 禁用外部触发中断。



TC 中断屏蔽寄存器

寄存器名称： TC_IMR

访问类型： 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ETRGS	LDRBS	LDRAS	CPCS	CPBS	CPAS	LOVRS	COVFS

- **COVFS: 计数器溢出**

0 = 计数器溢出中断禁用。

1 = 计数器溢出中断使能。

- **LOVRS: 载入溢出**

0 = 载入溢出中断禁用。

1 = 载入溢出中断使能。

- **CPAS: RA 比较**

0 = RA 比较中断禁用。

1 = RA 比较中断使能。

- **CPBS: RB 比较**

0 = RB 比较中断禁用。

1 = RB 比较中断使能。

- **CPCS: RC 比较**

0 = RC 比较中断禁用。

1 = RC 比较中断使能。

- **LDRAS: RA 载入**

0 = 载入 RA 中断禁用。

1 = 载入 RA 中断使能。

- **LDRBS: RB 载入**

0 = 载入 RB 中断禁用。

1 = 载入 RB 中断使能。

- **ETRGS: 外部触发**

0 = 外部触发中断禁用。

1 = 外部触发中断使能。

多媒体卡接口 (MCI)

概述

多媒体卡接口 (MCI) 支持多媒体卡 (MMC) 规范 V2.2 及 SD 存储卡规范 V1.0。

MCI 包括命令寄存器、响应寄存器、数据寄存器、暂停寄存器及错误检测逻辑，当需要时，它能自动处理命令发送，接收响应及有处理器开销限制的数据。

MCI 支持流、块与多块的数据读写，并与外设数据控制器通道兼容，大容量传输缓冲器可减小处理器干涉。

MCI的最高工作频率为主机时钟的2分频，并最多支持16个插槽连接(由产品决定)。每个插槽可用来与多媒体卡总线 (最大可连接 30 个卡) 或 SD 存储卡连接。一次只能选择一个插槽 (插槽可复用)。命令寄存器中有一位来执行该操作。

SD存储卡通信基于一个9脚接口(时钟、命令、四数据与三电源线)；多媒体卡基于一个7脚接口(时钟、命令、一数据与三电源线)。

SD 存储卡接口也支持多媒体卡操作。二者的主要不同在于初始化过程及总线布局。

MCI 主要特性如下：

- 与多媒体卡规范 V 2.2 兼容
- 与 SD 存储卡规范 V 1.0 兼容
- 卡时钟速率最高可达到主机时钟的 1/2
- 内置电源管理单元，可在不使用用时降低时钟速率
- 最多可支持 16 个复用插槽 (由产品决定)
  - 每条插槽可与一个多媒体卡总线 (最多可连接 30 个卡) 或一个 SD 存储卡连接
- 支持流、块及多块数据的读写
- 支持与外设数据控制器的连接
  - 大容量传输缓冲器使处理器干涉最小



方框图

Figure 233. 方框图

应用框图

Figure 234. 应用框图

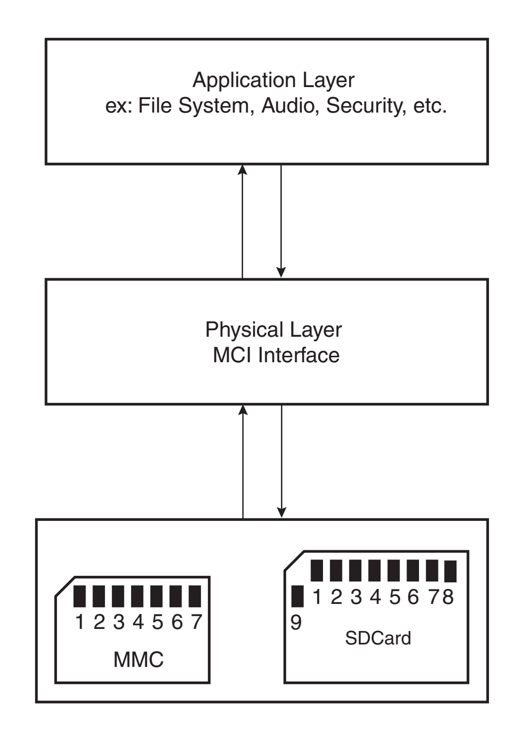


Table 93. I/O 线说明

引脚名称	引脚说明	类型 (1)	注释
MCCDA/MCCDB	命令 / 响应	I/O/PP/OD	MMC 或 SD 卡的 CMD
MCCK	时钟	I	MMC 或 SD 卡的 CLK
MCDA0 - MCDA3	插槽 A 数据 0..3	I/O/PP	MMC 的 DAT0 SD 卡的 DAT[0..3]
MCDB0 - MCDB3	插槽 B 数据 0..3	I/O/PP	MMC 的 DAT0 SD 卡的 DAT[0..3

Note: 1. I：输入、 O：输出、 PP：推 / 拉、 OD：开漏。

附属产品

I/O 线

用来连接多媒体卡或 SD 卡的引脚可与 PIO 线复用。必须先对 PIO 控制器编程将外设功能分配到 MCI 引脚上。

电源管理

MCI 可由电源管理控制器 (PMC) 定时，因此必须先配置 PMC 以使能 MCI 时钟。

中断

MCI 接口有一条与高级中断控制器 (AIC) 连接的中断线。

处理 MCI 中断请求需要在配置 MCI 前对 AIC 编程。

总线布局

Figure 235. 多媒体存储卡总线布局

多媒体卡通信基于一个 7 脚串行总线接口。它有三条通信线及四条供电线。

Table 94. 总线布局

引脚序号	名称	类型 (1)	说明	MCI 引脚名称
1	RSV	NC	未连接	
2	CMD	I/O/PP/OD	命令 / 响应	MCCDA/MCCDB
3	VSS1	S	电源地	VSS
4	VDD	S	电源电压	VDD
5	CLK	I	时钟	MCCK
6	VSS2	S	电源地	VSS
7	DAT[0]	I/O/PP	数据 0	MCDA0/MCDB0

Note: 1. I：输入、 O：输出、 PP：推 / 拉、 OD：开漏。

Figure 236. MMC 总线连接

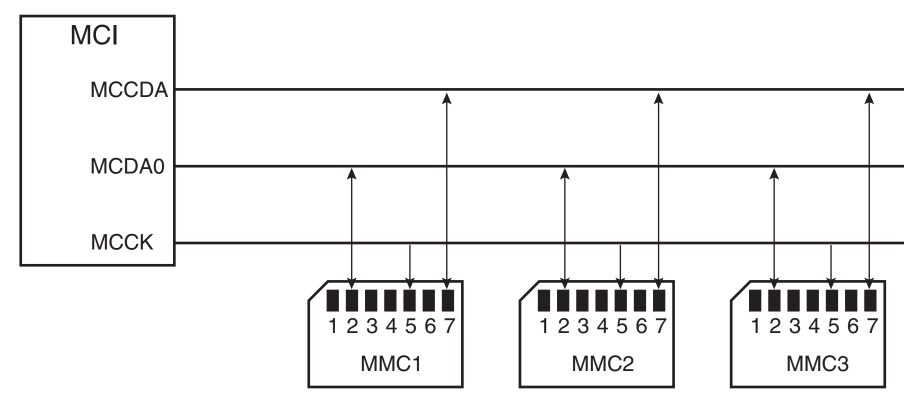
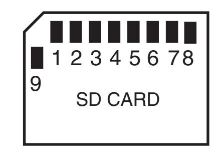


Figure 237. SD 存储卡总线连接



SD 存储卡总线信号列于 Table 95。

Table 95. SD Memory Card Bus Signals

引脚序号	名称	类型 (1)	说明	MCI 引脚名称
1	CD/DAT[3]	I/O/PP	卡检测 / 数据线位 3	MCDA3/MCDB3
2	CMD	PP	命令 / 响应	MCCDA/MCCDB
3	VSS1	S	电源地	VSS
4	VDD	S	电源电压	VDD
5	CLK	I	时钟	MCCK
6	VSS2	S	电源地	VSS
7	DAT[0]	I/O/PP	数据线位 0	MCDA0/MCDB0
8	DAT[1]	I/O/PP	数据线位 1	MCDA1/MCDB1
9	DAT[2]	I/O/PP	数据线位 2	MCDA2/MCDB2

Note: 1. I : 输入、 O : 输出、 PP : 推 / 拉、 OD : 开漏。

Figure 238. SD 卡总线连接

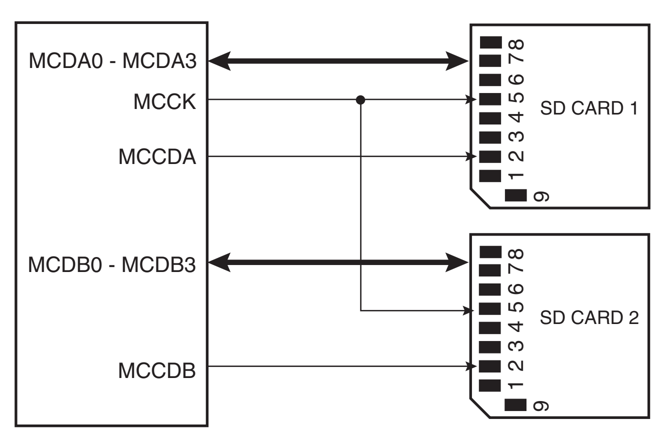
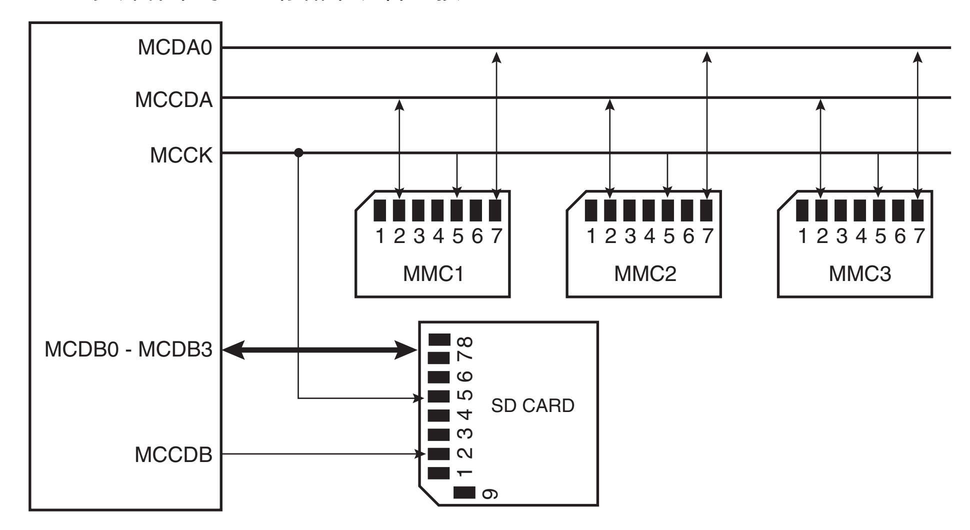




Figure 239. 多媒体卡与 SD 存储卡混合连接



当MCI配置为SD存储卡操作时，数据总线宽度由MCI_SDCR寄存器选择。对该寄存器SDCBUS位清零表示宽度为一位；若对该位置位则表示宽度为四位。对于多媒体卡操作，只可使用数据线 0。其它数据线可作为独立 PIO 使用。

多媒体卡操作

上电复位后，一个专用的基于多媒体卡总线协议的信息将卡初始化。每条信息由以下任一记号表示：

- 命令:命令用来启动操作。命令可由主机发送到单卡上 (寻址命令) 或所有连接的卡上 (广播命令)。命令在 CMD 线上串行发送。
- 响应：响应是由定址卡或 (同步) 从所有连接的卡上向主机就前面收到的命令做出回答。响应在 CMD 线上串行发送。
- 数据：数据可在卡与主机间传输。数据提供数据线传输。

在初始化阶段由总线控制器对当前连接的卡进行地址分配，实现卡定址。每个卡均有一个唯一的 CID 序号。

命令、响应及数据块结构见多媒体卡系统规范 V 2.2， Table 96 on page 501 中也有说明。

多媒体卡数据传输由这些内容组成。

多媒体卡有不同的操作类型。定址操作通常包括一条命令及一个响应。此外，某些操作中包含数据；还有一些操作直接在命令或响应结构中传输数据。此时，操作中不出现数据记号。 DAT 与 CMD 线上的位以时钟 MCCK 同步传输。

定义以下两类数据传输：

- 序列命令：这些命令初始化一个连续数据流。只有当 CMD 线后出现停止命令时终止。该模式将命令开销降到一个极小的范围。
- 块定向命令：这些命令由 CRC 位连续发送数据块。

读、写操作均允许单或多块传输。与序列读类似，当 CMD 线后出现停止命令时终止多块传输。

MCI 提供一套寄存器来执行所有的多媒体卡操作。

命令 - 响应操作

复位后 MCI 禁用，只有当 MCI_CR 控制寄存器的 MCIEN 位置位后才重新有效。 PWSEN 位通过将 MCI 时钟进行 2 的 PWSDIV 次幂分频，以降低功耗。

卡的命令与响应由 MCCK 上升沿计时。

所有多媒体卡时间均定义在多媒体卡系统规范 V 2.2 中。

需要两种总线模式 (开漏与推 / 拉) 来处理定义在 MCI 命令寄存器中的操作。MCI_CMDR 允许执行命令。

例如，执行 ALL_SEND_CID 命令：

	主机命令				N_{ID} 周期				CID 或 OCR			
CMD	S	T	内容	CRC	E	Z	******	Z	S	T	Content	Z	Z	Z

Table 96 与 Table 97 给出命令 ALL_SEND_CID 及 MCI_CMDR 控制寄存器的域及值。

Table 96. ALL_SEND_CID 命令说明

CMD 索引	类型	变量	Resp	缩写	命令说明
CMD2	bcr	[31:0] 填充位	R2	ALL_SEND_CID	要求所有卡在 CMD 线上发送它们的 CID 序号

Table 97. MCI_CMDR 命令寄存器域与值

域	值
CMDNB (命令序号)	2 (CMD2)
RSPTYP (响应类型)	2 (R2：136 位响应)
SPCMD (专用命令)	0 (不是专用命令)
OPCMD (开漏命令)	1
MAXLAT (命令到响应的最大等待时间)	0 (NID 周期 ==> 5 周期)
TRCMD (传输命令)	0 (无传输)
TRDIR (传输方向)	X (只在传输命令中有效)
TRTYP (传输类型)	X (只在传输命令中有效)

MCI_ARGR 包含了命令的变量域。

发送一条命令，必须执行下列步骤：

- 将命令变量填入变量寄存器 (MCI_ARGR)。
- 设置命令寄存器 (MCI_CMDR) (见 Table 97)。

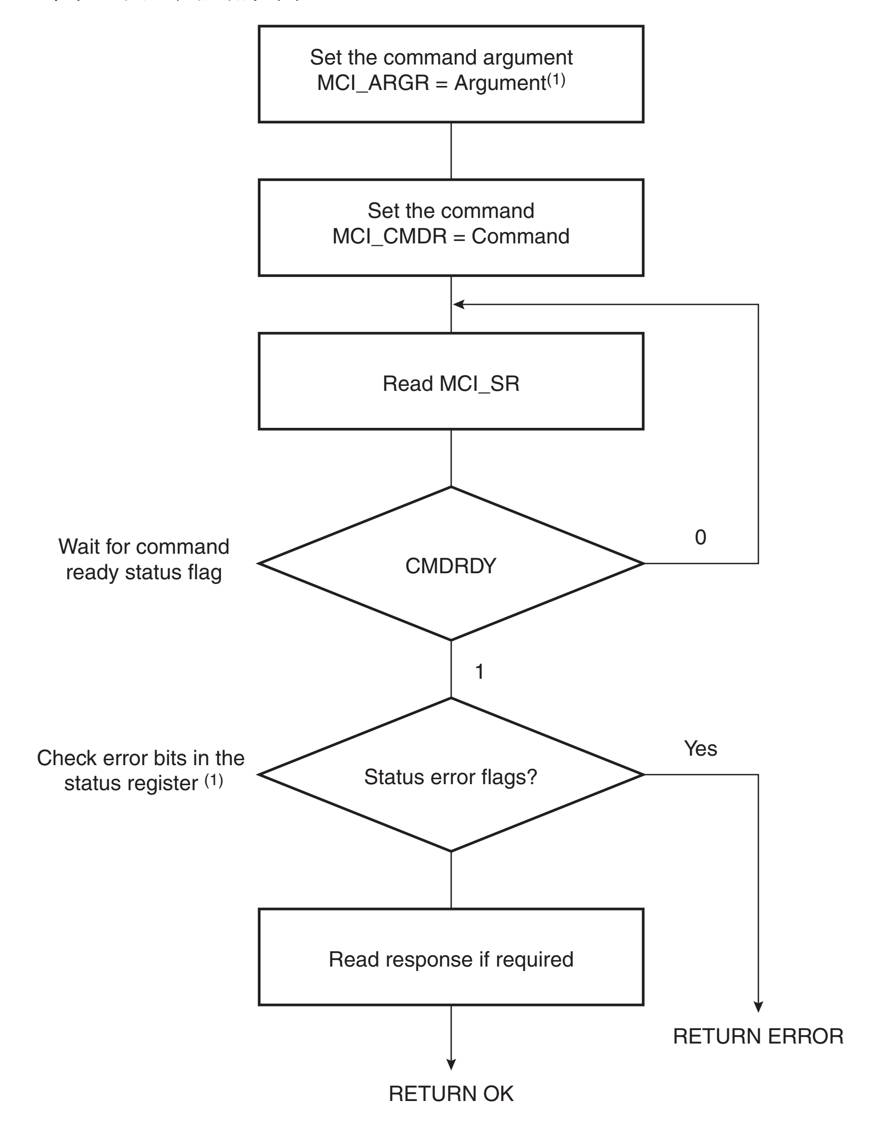
在写命令寄存器后立即发送命令。状态寄存器 (MCI_SR) 的 CMDRDY 状态位在命令结束后出现。若命令需要响应，可在 MCI 响应寄存器 (MCI_RSPR) 中读取。根据命令不同，响应大小从 48 位到 136 位。 MCI 内置一个错误检测以防止传输中对数据的破坏。

下面的流程图给出如何向卡发送命令及在需要时读响应。本例中，对状态寄存器位进行轮询，但要设置中断使能寄存器 (MCI_IER) 中相应的位以允许使用中断方式。





Figure 240. 命令 / 响应功能流程图



Note: 1. 若命令为 SEND_OP_COND， CRC 错误标志始终存在 (参见多媒体卡规范中的 R3 响应)。

数据传输操作

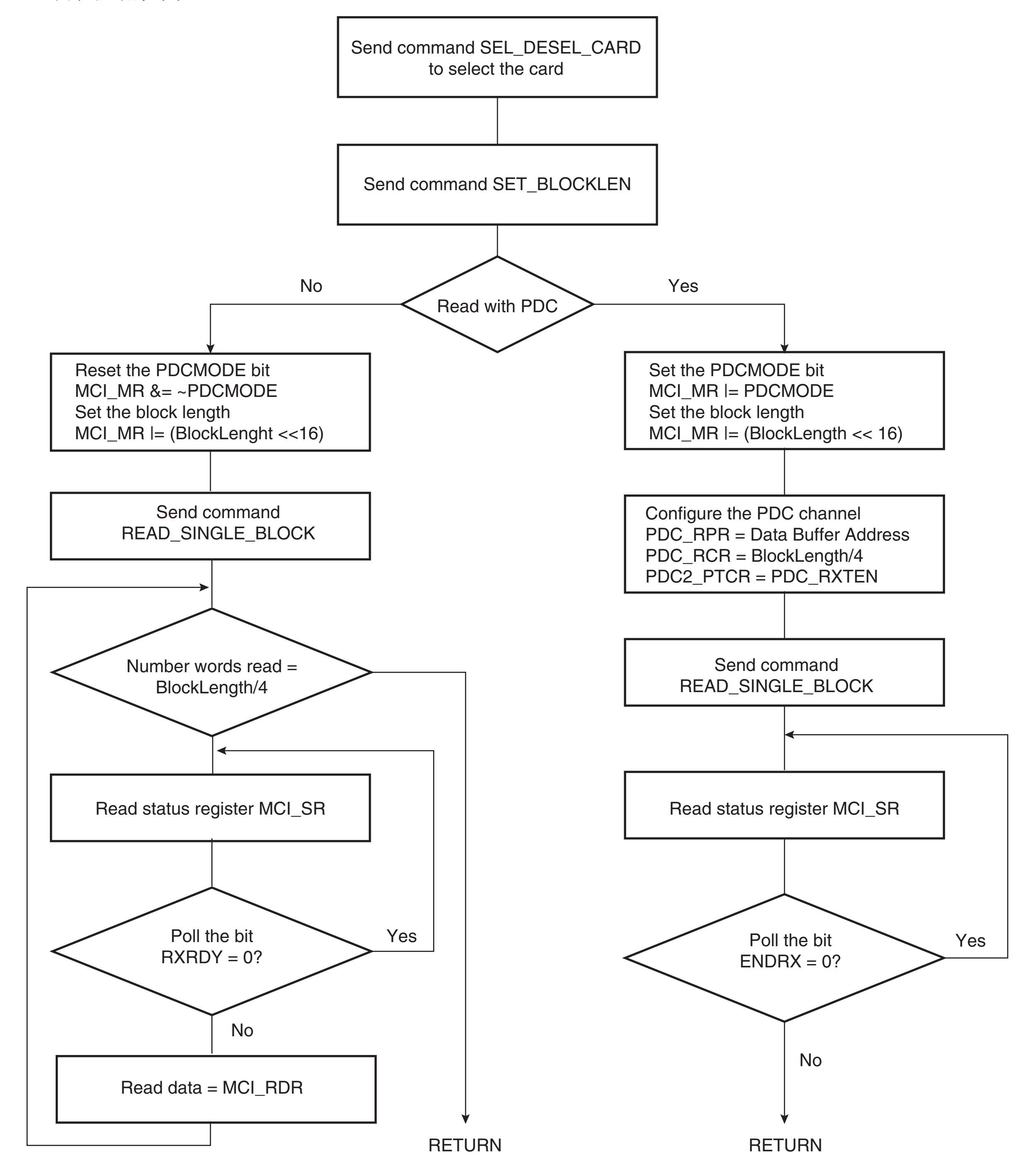
多媒体卡允许多种读 / 写操作 (单块、多块、流等)。

使用外设数据控制器 (PDC) 特性进行这些操作。若 MCI_MR 中 PDCMODE 位置位，则所有的读写均使用 PDC 工具。所有情况下，块长度必须在模式寄存器中定义。

读操作

以下流程图给出用或不用 PDC 工具对单块的读取。本例中，使用轮询方式来等待读结束。类似的，用户可配置中断使能寄存器 (MCI_IER) 在读结束后触发中断。这两种方法适用于所有多媒体卡读操作。

Figure 241. 读功能流程图



写操作

写操作中， MCI 模式寄存器 (MCI_MR) 用来定义当写非多块时的填充值。若 PDCPADV 位为 0，当填充数据时，值为 0x00 ；否则使用 0xFF。若设置， PDCMODE 位使能 PDC 传输。

以下流程图给出用或不用 PDC 工具对单块的写入。根据中断屏蔽寄存器 (MCI_IMR) 中的内容，可使用轮询或中断方式来等待写结束。

该流程图适用于所有多媒体卡的写操作。

Figure 242. 写功能流程图

SD 卡操作

多媒体卡接口能执行 SD 存储卡 (安全数字存储卡) 命令。 SD 存储卡包括版权保护机制以达到 SDMI 标准的安全要求，适应高速大容量存储器。

物理形状因数，引脚分配及数据传输协议与多媒体卡基本相同，只是附加一些要求。

SD存储卡通信基于9引脚接口(时钟、命令、4 x数据及3 x电源线)。规范中定义了通信协议。SD 存储卡与多媒体卡的最大不同在于初始化过程。

SD 卡控制寄存器 (MCI_SDCR) 允许卡插槽及数据宽度的选择。

SD 卡总线允许动态配置数据线的条数。上电后 ,默认 SD 存储卡只使用 DAT0 作为数据传输。初始化后，主机可改变总线宽度 (激活数据线数)。



多媒体卡 (MCI) 用户接口

Table 98. MCI 寄存器映射

偏移	寄存器	寄存器名称	访问类型	复位值
0x00	控制寄存器	MCI_CR	只写	---
0x04	模式寄存器	MCI_MR	读 / 写	0x0
0x08	数据暂停寄存器	MCI_DTOR	读 / 写	0x0
0x0C	SD 卡寄存器	MCI_SDCR	读 / 写	0x0
0x10	变量寄存器	MCI_ARGR	读 / 写	0x0
0x14	命令寄存器	MCI_CMDR	只写	---
0x18 - 0x1C	保留			
0x20	响应寄存器 (1)	MCI_RSPR	只读	0x0
0x24	响应寄存器 (1)	MCI_RSPR	只读	0x0
0x28	响应寄存器 (1)	MCI_RSPR	只读	0x0
0x2C	响应寄存器 (1)	MCI_RSPR	只读	0x0
0x30	接收数据寄存器	MCI_RDR	只读	0x0
0x34	发送数据寄存器	MCI_TDR	只写	---
0x38 - 0x3C	保留			
0x40	状态寄存器	MCI_SR	只读	0xC0E5
0x44	中断使能寄存器	MCI_IER	只写	---
0x48	中断禁用寄存器	MCI_IDR	只写	---
0x4C	中断屏蔽寄存器	MCI_IMR	只读	0x0
0x50-0xFF	保留			
0x100-0x124	保留给 PDC			

Note: 1. 响应寄存器可由 MCI_RSPR 读 N 次或在连续地址 (0x20 到 0x2C)。
N 由响应大小决定。

MCI 控制寄存器

寄存器名称：MCI_CR

访问类型： 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	–	–	PWSDIS	PWSEN	MCIDIS	MCIEN

- **MCIEN: 多媒体接口使能**

0 = 无效。

1 = 若 MCDIS 为 0，使能多媒体接口。

- **MCIDIS: 多媒体接口禁用**

0 = 无效。

1 = 禁用多媒体接口。

- **PWSEN: 节能模式使能**

0 = 无效。

1 = 若 PWSDIS 为 0，使能节能模式。

- **PWSDIS: 节能模式禁用**

0 = 无效。

1 = 禁用节能模式。



MCI 模式寄存器

寄存器名称： MCI_MR

访问类型： 读 / 写

31	30	29	28	27	26	25	24
–	–	BLKLEN					

23	22	21	20	19	18	17	16
BLKLEN						0	0

15	14	13	12	11	10	9	8
PDCMODE	PDCPADV	–	–	–	PWSDIV		

7	6	5	4	3	2	1	0
CLKDIV							

- **CLKDIV: 时钟分频器**

多媒体卡接口时钟 (MCCK) 为主机时钟 (MCK) 由 (2*(CLKDIV+1)) 分频。

- **PWSDIV: 节能分频器**

进入节能模式，多媒体卡接口时钟由 $2^{(PWSDIV)}$ 分频。

- **PDCPADV: PDC 填充值**

0 = 当填充数据到写传输时，使用 0x00(不只是 PDC 传输)。

1 = 当填充数据到写传输时，使用 0xFF(不只是 PDC 传输)。

- **PDCMODE: PDC 定向模式**

0 = 禁用 PDC 传输

1 = 使能 PDC 传输。此时 UNRE 与 OVRE (MCI_SR) 无效。

- **BLKLEN: 数据块长**

该域定义数据块大小。

位 16 与 17 必须为 0。

MCI 数据暂停寄存器

寄存器名称 : MCI_DTOR

访问类型 : 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	DTOMUL			DTOCYC			

- **DTOCYC: 数据暂停周期数**
- **DTOMUL: 数据暂停乘数**

这些域确定两数据块传输间最大主机时钟周期数。它等于 (DTOCYC x 乘数)。

乘数由 DTOMUL 定义，如下表所示 :

DTOMUL			乘数
0	0	0	1
0	0	1	16
0	1	0	128
0	1	1	256
1	0	0	1024
1	0	1	4096
1	1	0	65536
1	1	1	1048576

MCI SD 卡寄存器

寄存器名称 : MCI_SDCR

访问类型 : 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SDCBUS	–	–	–	SDCSEL			

- **SDCSEL: SD 卡选择**

0 = 选择 SD 卡 A。

1 = 选择 SD 卡 B。

- **SDCBUS**

0 = 1 位数据总线。

1 = 4 位数据总线。

MCI 变量寄存器

寄存器名称 : MCI_ARGR

访问类型 : 读 / 写

31	30	29	28	27	26	25	24
ARG							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
ARG							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
ARG							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ARG							

- **ARG: 命令变量**

MCI 命令寄存器

寄存器名称： MCI_CMDR

访问类型： 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	TRTYPE	TRDIR	TRCMD	

15	14	13	12	11	10	9	8
–	–	–	MAXLAT	OPDCMD	SPCMD		

7	6	5	4	3	2	1	0
RSPTYP		CMDNB					

当 MCI_SR 中 CMDRDY 为 0 且无中断命令发送 (位 SPCMD) 时，该寄存器写保护。即当前执行命令不能被中断或修改。

- **CMDNB: 命令数**
- **RSPTYP: 响应类型**

RSP		响应类型
0	0	无响应
0	1	48 位响应
1	0	136 位响应
1	1	保留

- **SPCMD: 专用 CMD**

SPCMD			CMD
0	0	0	无专用 CMD
0	0	1	初始化 CMD： 初始化序列须 74 个时钟周期
0	1	0	同步 CMD： 在发送挂起命令前等待当前数据块传输结束
0	1	1	保留
1	0	0	中断命令： 对应中断模式 (CMD40)
1	0	1	中断响应： 对应中断模式 (CMD40)

- **OPDCMD: 开漏命令**

0 = 推拉命令。

1 = 开漏命令。



- **MAXLAT: 命令到响应最大等待时间**

0 =5 周期最大等待。

1 = 64 周期最大等待。

- **TRCMD: 传输命令**

TRCMD		传输类型
0	0	无传输
0	1	开始传输
1	0	停止传输
1	1	保留

- **TRDIR: 传输方向**

0 = 写。

1 = 读。

- **TRTYP: 传输类型**

TRTYP		传输类型
0	0	块
0	1	多块
1	0	流
1	1	保留

MCI SD 响应寄存器

寄存器名称 : MCI_RSPR

访问类型 : 只读

31	30	29	28	27	26	25	24
RSP							

23	22	21	20	19	18	17	16
RSP							

15	14	13	12	11	10	9	8
RSP							

7	6	5	4	3	2	1	0
RSP							

- **RSP: 响应**

MCI SD 接收数据寄存器

寄存器名称 : MCI_RDR

访问类型 : 只读

31	30	29	28	27	26	25	24
DATA							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
DATA							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
DATA							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DATA							

- **DATA: 读取数据**

MCI SD 发送数据寄存器

寄存器名称 : MCI_TDR

访问类型 : 只写

31	30	29	28	27	26	25	24
DATA							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
DATA							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
DATA							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DATA							

- **DATA: 写数据**



MCI 状态寄存器

寄存器名称： MCI_SR

访问类型： 只读

31	30	29	28	27	26	25	24
UNRE	OVRE	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	DTOE	TCRCE	RTOE	RENDE	RCRCE	RDIRE	RINDE
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
TXBUFE	RXBUFF	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ENDTX	ENDRX	NOTBUSY	DTIP	BLKE	TXRDY	RXRDY	CMDRDY

- **CMDRDY: 命令就绪**

0 = 命令正在处理。

1 = 已发送最后一条命令。当在 MCI_CMDR 写入时清零。

- **RXRDY: 接收器就绪**

0 = 上次读 MCI_RDR 后未接收到数据。

1 = 上次读 MCI_RDR 后已接收到数据。

- **TXRDY: 发送就绪**

0= 最后写入 MCI_TDR 中的数据还未发送到移位寄存器。

1= 最后写入 MCI_TDR 中的数据已发送到移位寄存器。

- **BLKE: 数据块结束**

0 = 数据块发送未结束。

1 = 数据块发送已结束。在最后块结束时在 PDCMODE 中置位，或者在第一个块结束时置位。当读 MCI_SR 时清零。

- **DTIP: 数据传输正在进行**

0 = 没有数据传输处理进行。

1 = 当前数据传输正在进行，包括 CRC16 计算。在 CRC16 计算结束后清零。

- **NOTBUSY: 数据不忙**

0 = 卡未准备好接收新数据。

1 = 卡已准备好接收新数据 (对应卡空数据接收缓冲器的数据线 DAT0 高)。

- **ENDRX: RX 缓冲器结束**

0 = 上次写 MCI_RCR 或 MCI_RNCR 后接收计数寄存器未达到 0。

1 = 上次写 MCI_RCR 或 MCI_RNCR 后接收计数寄存器已达到 0。

- **ENDTX: TX 缓冲器结束**

0 = 上次写 MCI_TCR 或 MCI_TNCR 后发送计数寄存器未达到 0。

1 = 上次写 MCI_TCR 或 MCI_TNCR 后发送计数寄存器已达到 0。

- **RXBUFF: RX 缓冲器满**

0 = MCI_RCR 或 MCI_RNCR 为非 0 值。

1 = MCI_RCR 与 MCI_RNCR 中值均为 0。

- **TXBUFE: TX 缓冲器空**

0 = MCI_TCR 或 MCI_TNCR 为非 0 值。

1 = MCI_TCR 与 MCI_TNCR 中值均为 0。

- **RINDE: 响应索引错误**

0 = 无错误。

1 = 检测到发送命令索引与接收响应索引不匹配。当写 MCI_CMDR 时清零。

- **RDIRE: 响应方向错误**

0 = 无错误。

1 = 响应中由卡到主机方向位未检测到。

- **RCRCE: 响应 CRC 错误**

0 = 无错误。

1 = 响应中检测到 CRC7 错误。当写入 MCI_CMDR 时清零。

- **RENDE: 响应结束位错误**

0 = 无错误。

1 = 未检测到响应结束位。当写入 MCI_CMDR 时清零。

- **RTOE: 响应超时错误**

0 = 无错误。

1 = 已超出 MCI_CMDR 寄存器 MAXLAT 中设置的响应超时。当写入 MCI_CMDR 时清零。

- **DCRCE: 数据 CRC 错误**

0 = 无错误。

1 = 上次读最后数据块时，检测到 CRC16 错误。当发送新数据传输命令时清零。

- **DTOE: 数据超时错误**

0 = 无错误。

1 = 已超出 MCI_DTOR 寄存器 DTOCYC 与 DTOMUL 中设置的响应超时。当写入 MCI_CMDR 时清零。

- **OVRE: 溢出**

0 = 无错误。

1 = 最少丢失一个 8 位接收数据。当发送新数据传输命令时清零。

- **UNRE: 欠载运行**

0 = 无错误。

1 = 最少发送一个无有效信息的 8 位数据 (不写入)。当发送新数据传输命令时清零。



MCI 中断使能寄存器

寄存器名称 : MCI_IER

访问类型 : 只写

31	30	29	28	27	26	25	24
UNRE	OVRE	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	DTOE	TCRCE	RTOE	RENDE	RCRCE	RDIRE	RINDE
15	14	13	12	11	10	9	8
TXBUFE	RXBUFF	–	–	–	–	–	–
7	6	5	4	3	2	1	0
ENDTX	ENDRX	NOTBUSY	DTIP	BLKE	TXRDY	RXRDY	CMDRDY

- **CMDRDY: 命令就绪中断使能**
- **RXRDY: 接收器就绪中断使能**
- **TXRDY: 发送就绪中断使能**
- **BLKE: 数据块结束中断使能**
- **DTIP: 数据正在发送中断使能**
- **NOTBUSY: 数据不忙中断使能**
- **ENDRX: 接收缓冲器结束中断使能**
- **ENDTX: 发送缓冲器结束中断使能**
- **RXBUFF: 接收缓冲器满中断使能**
- **TXBUFE: 发送缓冲器空中断使能**
- **RINDE: 响应索引错误中断使能**
- **RDIRE: 响应方向错误中断使能**
- **RCRCE: 响应 CRC 错误中断使能**
- **RENDE: 响应位错误结束中断使能**
- **RTOE: 响应超时错误中断使能**
- **DCRCE: 数据 CRC 错误中断使能**
- **DTOE: 数据超时错误中断使能**
- **OVRE: 溢出中断使能**
- **UNRE: 欠载运行中断使能**

0 = 无效。

1 = 使能相应中断。

MCI Interrupt Disable Register

寄存器名称： MCI_IDR

访问类型： 只写

31	30	29	28	27	26	25	24
UNRE	OVRE	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	DTOE	TCRCE	RTOE	RENDE	RCRCE	RDIRE	RINDE
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
TXBUFE	RXBUFF	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ENDTX	ENDRX	NOTBUSY	DTIP	BLKE	TXRDY	RXRDY	CMDRDY

- **CMDRDY: 命令就绪中断禁用**
- **RXRDY: 接收器就绪中断禁用**
- **TXRDY: 发送就绪中断禁用**
- **BLKE: 数据块结束中断禁用**
- **DTIP: 数据正在发送中断禁用**
- **NOTBUSY: 数据不忙中断禁用**
- **ENDRX: 接收缓冲器结束中断禁用**
- **ENDTX: 发送缓冲器结束中断禁用**
- **RXBUFF: 接收缓冲器满中断禁用**
- **TXBUFE: 发送缓冲器空中断禁用**
- **RINDE: 响应索引错误中断禁用**
- **RDIRE: 响应方向错误中断禁用**
- **RCRCE: 响应 CRC 错误中断禁用**
- **RENDE: 响应位错误结束中断禁用**
- **RTOE: 响应超时错误中断禁用**
- **DCRCE: 数据 CRC 错误中断禁用**
- **DTOE: 数据超时错误中断禁用**
- **OVRE: 溢出中断禁用**
- **UNRE: 欠载运行中断禁用**

0 = 无效。

1 = 禁用相应中断。



MCI 中断屏蔽寄存器

寄存器名称 : MCI_IMR

访问类型 : 只读

31	30	29	28	27	26	25	24
UNRE	OVRE	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	DTOE	TCRCE	RTOE	RENDE	RCRCE	RDIRE	RINDE
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
TXBUFE	RXBUFF	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ENDTX	ENDRX	NOTBUSY	DTIP	BLKE	TXRDY	RXRDY	CMDRDY

- **CMDRDY: 命令就绪中断屏蔽**
- **RXRDY: 接收器就绪中断屏蔽**
- **TXRDY: 发送就绪中断屏蔽**
- **BLKE: 数据块结束中断屏蔽**
- **DTIP: 数据正在发送中断屏蔽**
- **NOTBUSY: 数据不忙中断屏蔽**
- **ENDRX: 接收缓冲器结束中断屏蔽**
- **ENDTX: 发送缓冲器结束中断屏蔽**
- **RXBUFF: 接收缓冲器满中断屏蔽**
- **TXBUFE: 发送缓冲器空中断屏蔽**
- **RINDE: 响应索引错误中断屏蔽**
- **RDIRE: 响应方向错误中断屏蔽**
- **RCRCE: 响应 CRC 错误中断屏蔽**
- **RENDE: 响应位错误结束中断屏蔽**
- **RTOE: 响应超时错误中断屏蔽**
- **DCRCE: 数据 CRC 错误中断屏蔽**
- **DTOE: 数据超时错误中断屏蔽**
- **OVRE: 溢出中断屏蔽**
- **UNRE: 欠载运行中断屏蔽**

0 = 相应中断未使能。

1 = 相应中断使能。

USB 器件端口 (UDP)

概述

USB器件端口(UDP)适用于通用串行总线(USB) V2.0全速器件规范。它为Atmel的与ARM7TDMI 与 ARM9TDMI 内核连接的内置 USB 收发器设计。

每个端点可配置为几种 USB 传输类型中的一种。它可与双端 RAM 的一段或两段联合用来存储当前数据有效负载。若使用两段，一个 DPR 段由处理器读、写，另外一个 DPR 段则由 USB 器件外设读、写。对于同步端点强制使用该特性。因此器件工作于有两 DPR 段端点时，保持最大带宽 (1M 字节 /s)。

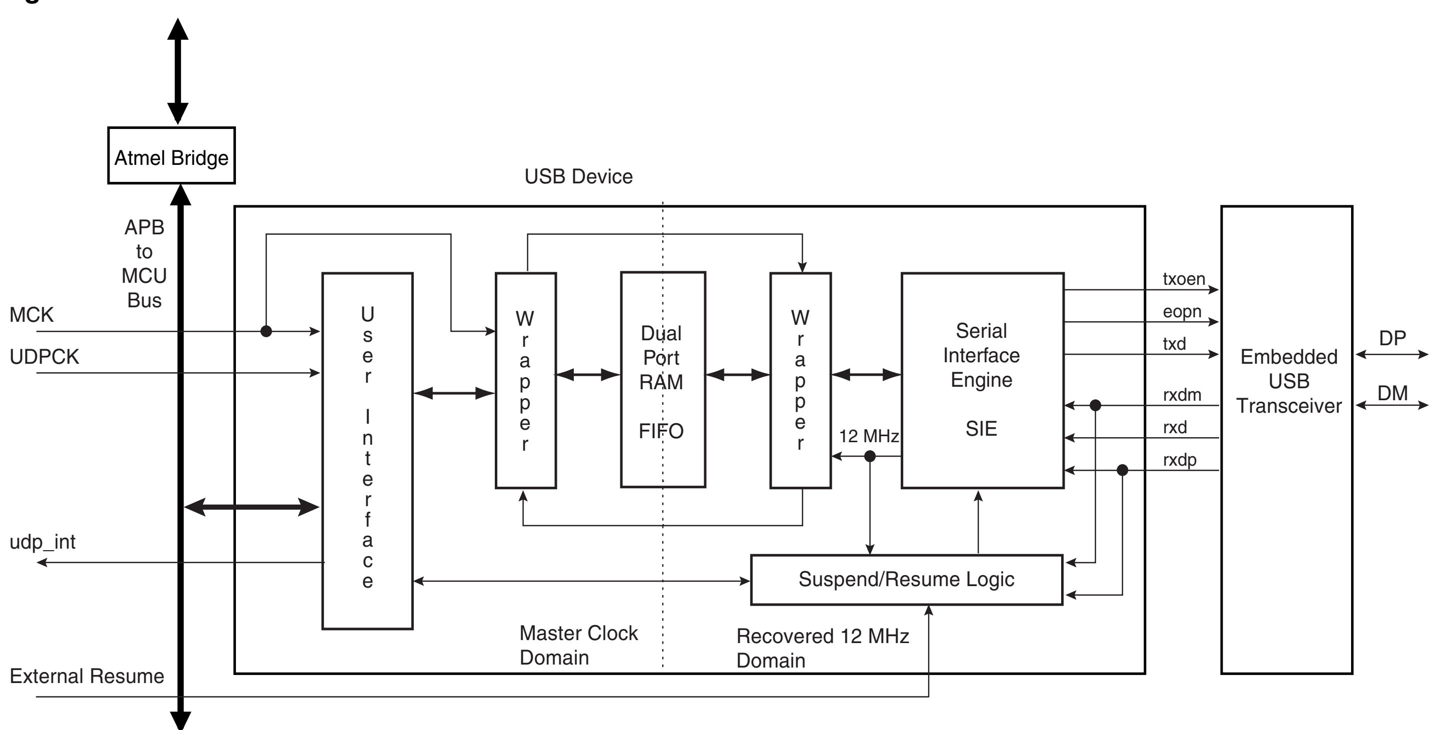
USB 器件自动检测挂起与恢复，通过出现中断来停止处理器。某些产品中可利用外部信号唤醒 USB 主机控制器发送。

UDP 主要特性如下：

- 与 USB V2.0 完全兼容， 12 Mb/s
- 内置 USB V2.0 全速收发器
- RTL 中端点数目与大小全参数化
- 端点内置双端 RAM
- 挂起 / 恢复逻辑
- 用于同步与批端点的 Ping-pong 模式 (2 存储器组)

方框图

Figure 243. USB 器件端口框图



通过 APB 总线接口访问 UDP。通过对 APB 寄存器 8 位值读写来对数据 FIFO 进行读写。

UDP 外设需要两个时钟：用于 MCK 域的外设时钟及用于 12 MHz 域的 48 MHz 时钟。





USB 2.0 全速垫内置并由串行接口引擎 (SIE) 控制。

external_resume 信号可选。它允许在系统模式下唤醒 UDP 外设。然后主机将通知请求恢复的器件。列举时，该特性必须由主机处理。

附属产品

USB 物理收发器集成在产品中。双向差分信号 DP 与 DM 对于产品边界有效。

应用中可能会用到两个 I/O 线：

- 一条检查来自主机的 VBUS 是否仍然有效。可能使用该入口通知自供电器件主机断电。此时，禁用板上上拉 DP 以防止电流流入主机。
- 一条用来控制板上上拉 DP。因此，当器件准备与主机通信时，通过该控制线激活其 DP 上拉。

I/O 线

PIO 控制器不控制 DP 与 DM。内置的 USB 物理收发器由 USB 器件外设控制。

为保留检查 VBUS 的 I/O 线，必须先对 PIO 控制器编程，将该 I/O 配置为输入 PIO 模式。

为保留板上拉控制 I/O 线，必须先对 PIO 控制器编程，将该 I/O 配置为输出 PIO 模式。

电源管理

USB 器件外设需要 48 MHz 时钟。该时钟由精度为 ± 0.25% 的 PLL 产生。

因此，USB 器件收到来自电源管理控制器 (PMC) 的两个时钟：主机时钟 MCK 用来驱动外设用户接口； UDPCK 用来与总线 USB 信号连接 (恢复的 12 MHz 域)。

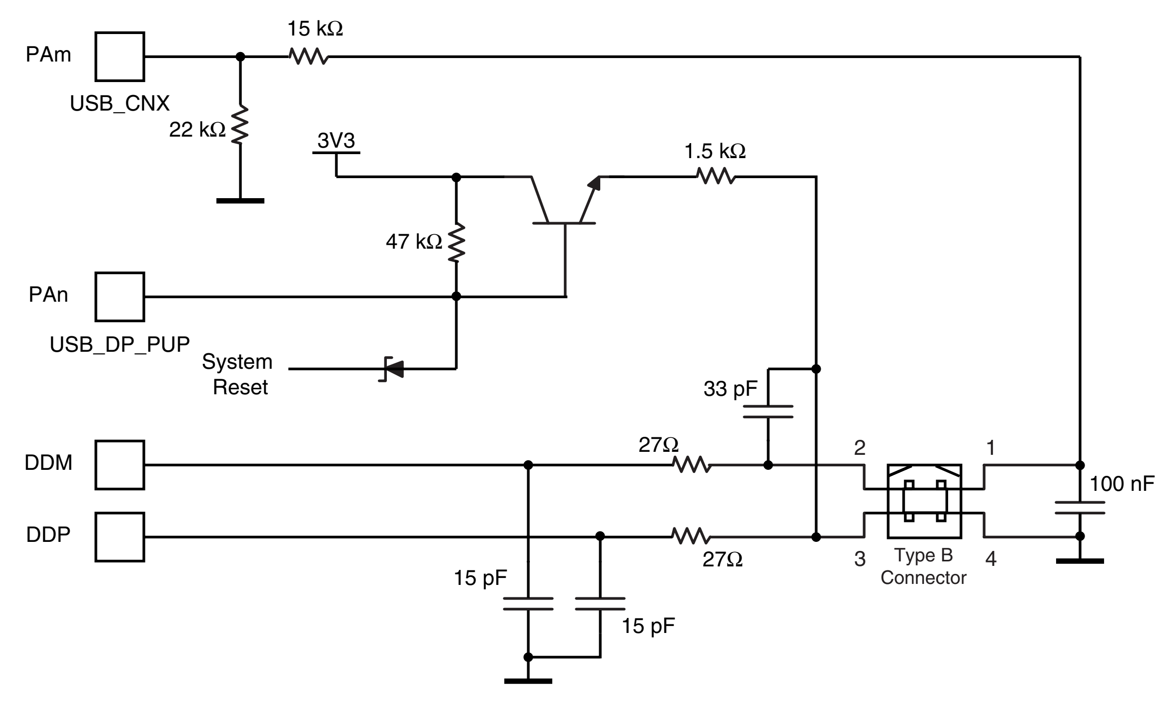
中断

USB 器件有一条与高级中断控制器 (AIC) 连接的中断线。

处理 USB 器件中断须在配置 UDP 前对 AIC 编程。

典型连接

Figure 244. 与 USB 器件外设连接示意图



输入信号 USB_CNX 用来检验主机是否连接。

输出信号 USB_DP_PUP 用来使能 DP 上拉。

Figure 244 给出复位后上拉自动激活示意图。

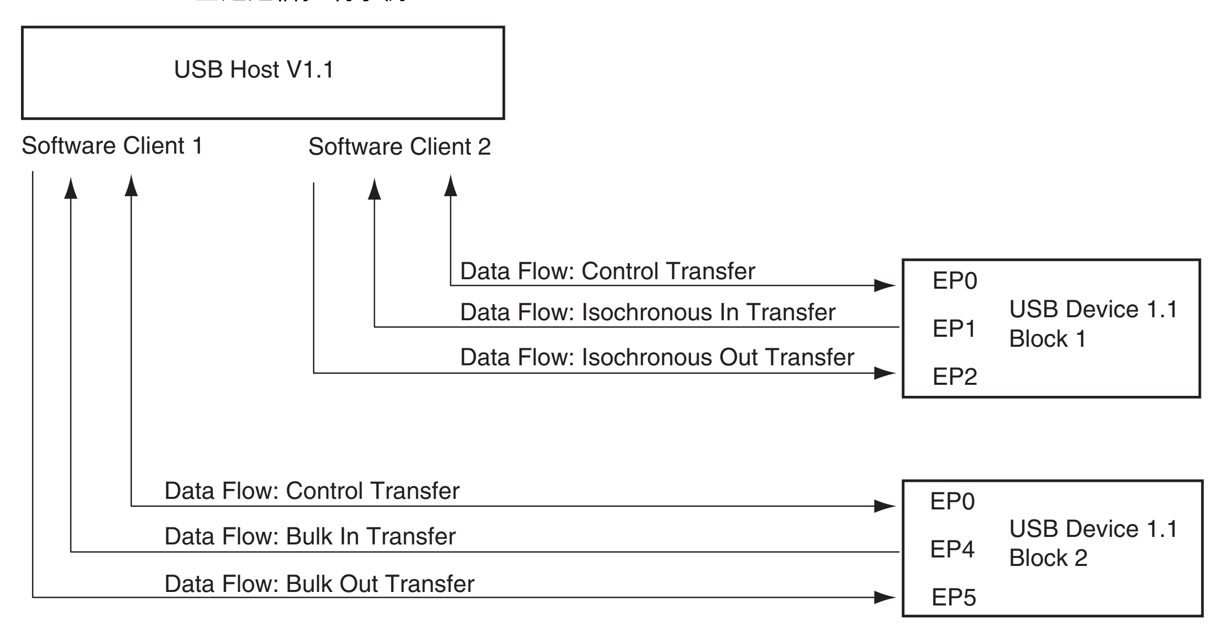




功能说明

USB V2.0 全速介绍 USB V2.0 全速提供主机与附属 USB 器件间的通信服务。每个器件提供与每个端点相关的通信流。软件通过通信流实现主机与 USB 器件的通信。

Figure 245. USB V2.0 全速通信控制示例



USB V2.0 全速传输类型

USB 器件定义四种传输类型。

Table 99. USB 通信流

传输	方向	带宽	端点大小	错误检测	重试
控制	双向	无保证	8, 16, 32, 64	有	自动
同步	单向	保证	1 - 1023	有	无
中断	单向	无保证	≤64	有	有
批	单向	无保证	8, 16, 32, 64	有	有

USB 总线处理

每次发送将在 USB 总线上进行一个或多个处理。共有五种处理类型：

1. 设置处理
2. 数据入处理
3. 数据出处理
4. 状态入处理
5. 状态出处理

USB 发送事件定义

如 Table 100 所示， USB 总线上发送顺序进行。

Table 100. USB 发送事件

控制发送 [1] [3]	• 设置处理 > 数据入处理 > 状态出处理 • 设置处理 > 数据出处理 > 状态入处理 • 设置处理 > 状态入处理
中断入发送 (器件向主机)	• 数据入处理 > 数据入处理
中断出发送 (主机向器件)	• 数据出处理 > 数据出处理
同步入发送 [2] (器件向主机)	• 数据入处理 > 数据入处理
同步出发送 [2] (主机向器件)	• 数据出处理 > 数据出处理
批入发送 (器件向主机)	• 数据入处理 > 数据入处理
批出发送 (主机向器件)	• 数据出处理 > 数据出处理

Notes: 1. 控制发送必须使用无 ping-pong 特性的端点。
2. 同步发送必须使用有 ping-pong 特性的端点。
3. 使用延迟握手可忽略控制发送。



与 USB V2.0 器件外设交互处理

设置处理

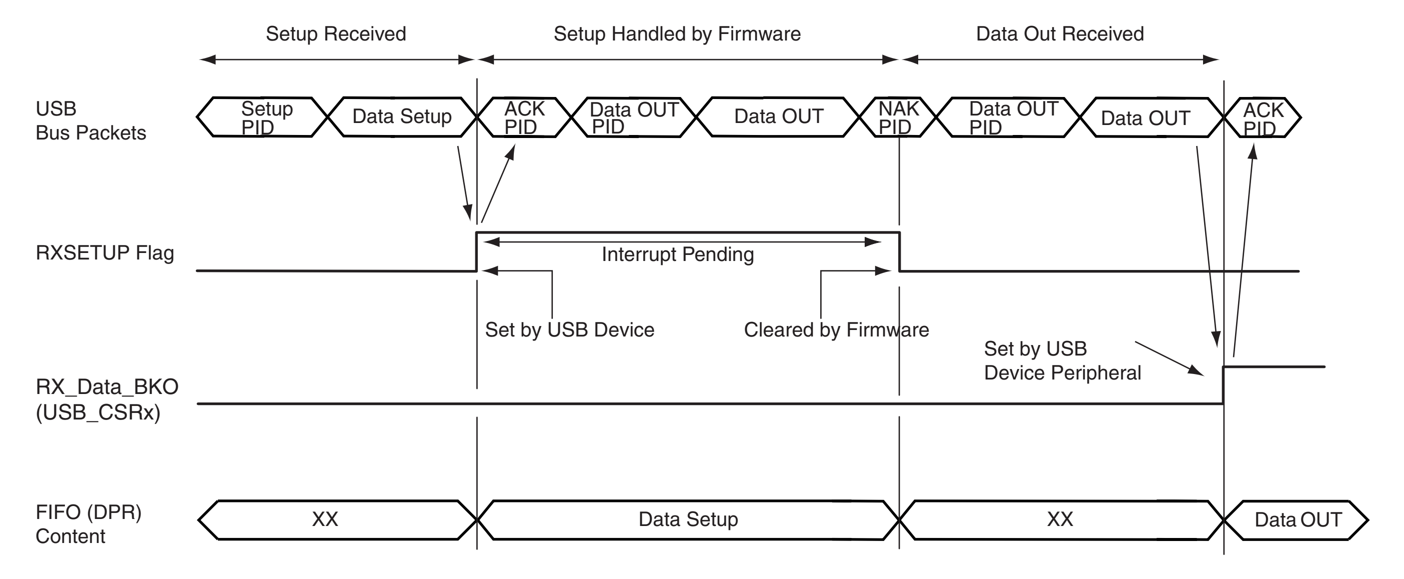
设置是控制发送时主机到器件处理的专用类型。控制发送必须使用无 ping-pong 特性的端点。设置处理应尽快由固件实现。它使用从主机到器件的发送请求。然后这些请求由 USB 器件处理，并可能需要更多的变量。变量由设置处理后的数据出处理发送到器件。这些处理也可能返回数据。这些数据由设置处理后的下一个数据入处理送入主机。状态处理结束整个控制发送。

当 USB 端点收到设置发送时：

- USB 器件自动应答该设置包。
- 设置 USB_CSRx 寄存器中的 RXSETUP。
- 若 RXSETUP 未被清零则产生一个端点中断。若该端点中断使能，该中断在微控制器上执行。

因此，固件必须轮询 USB_CSRx 以检测 RXSETUP 或获取中断，从 FIFO 中读设置包，然后清除 RXSETUP。在未读 FIFO 中数据包前， RXSETUP 不能清零。否则， USB 器件将接收下一个数据出发送并覆盖 FIFO 中设置包。

Figure 246. 设置处理及数据出处理



数据入处理

数据入处理用在控制、同步、批及中断处理中，引导数据由器件送往主机。同步发送中使用数据入处理的端点必须有 ping-pong 特性。

使用无 Ping-pong 特性端点

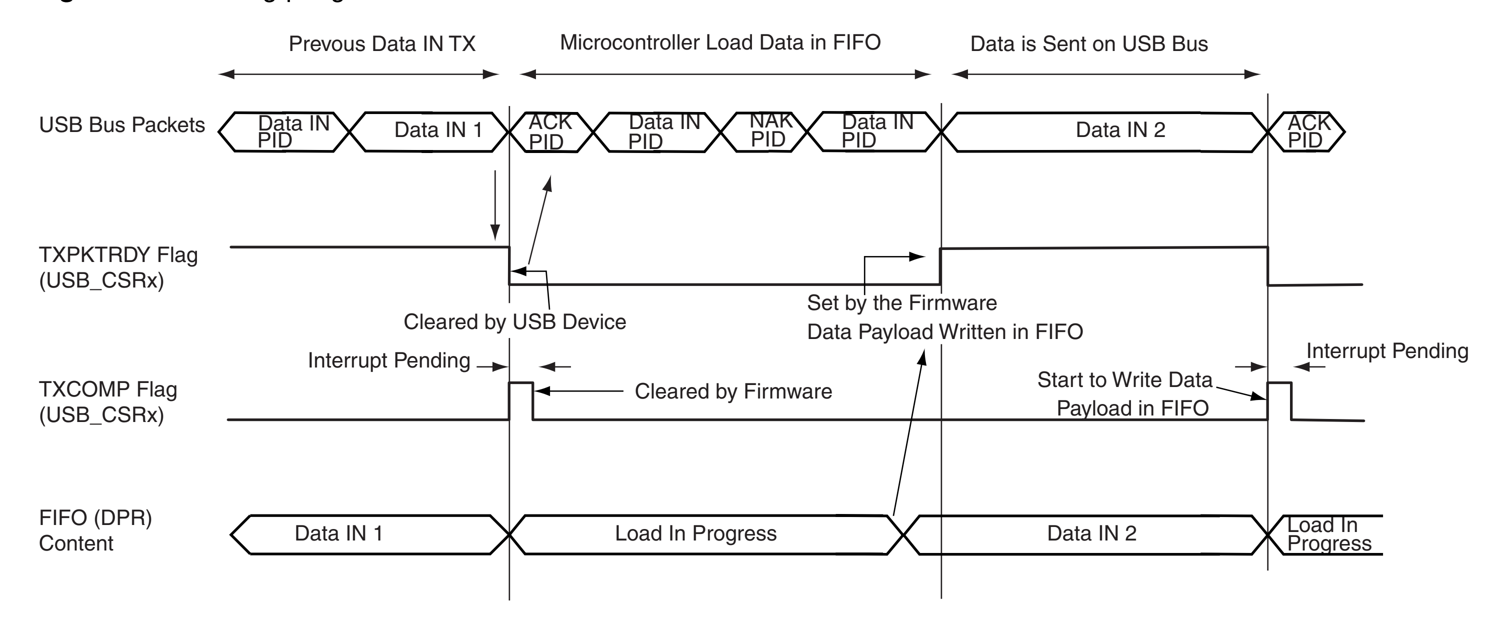
用无 ping-pong 端点执行数据入处理：

1. 微控制器检查是否通过轮询端点 USB_CSRx 寄存器的 TXPKTRDY 对 FIFO 写入 (TXPKTRDY 必须清零)。
2. 微控制器将准备发送的数据写入端点的 FIFO 中，给端点 USB_FDRx 寄存器写入 0 或多字节值。
3. 微控制器通过设置端点 USB_CSRx 寄存器的 TXPKTRDY 位来通知 USB 外设它已完成。
4. 当 USB_CSRx 寄存器已置位，微控制器通知已释放端点 FIFO。然后当 TXCOMP 置位时相应端点中断挂起。

当 USB 器件收到数据入包 ACK PID 信号时，将 TXCOMP 置位。当 TXCOMP 置位时中断挂起。

Note: 更多关于数据入协议层的内容请参见*通用串行总线规范*， *Rev 1.1* 的第八章。

Figure 247. 无 Ping-pong 端点的数据入发送



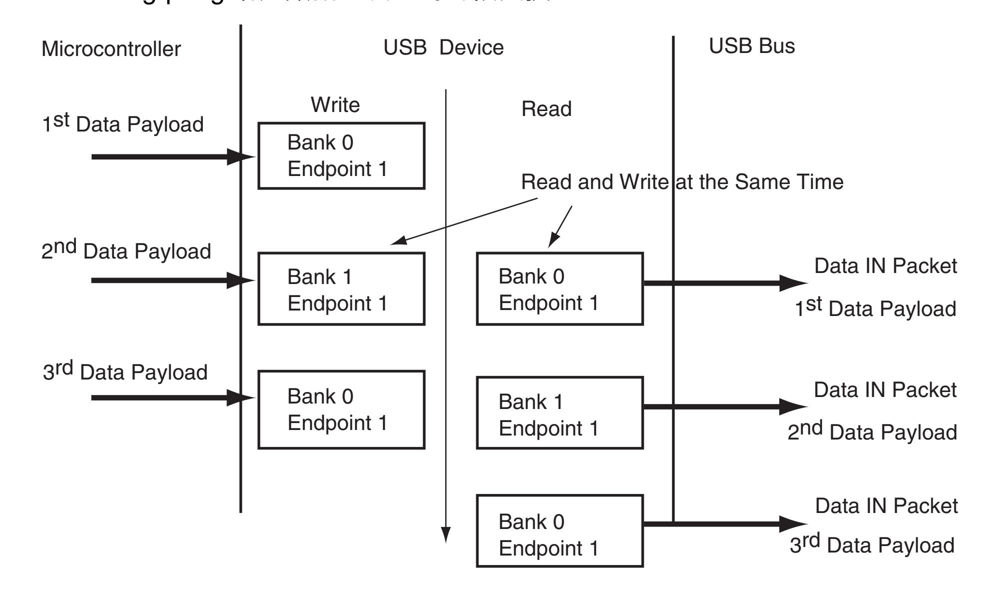




使用有 Ping-pong 特性端点

同步发送时需要使用有 ping-pong 特性的端点。为保证带宽为常数，当 USB 器件发送当前数据时，微控制器必须准备下一个数据有效负载。因此要使用两个存储器段。一个给微控制器使用，另一个由 USB 器件锁定。

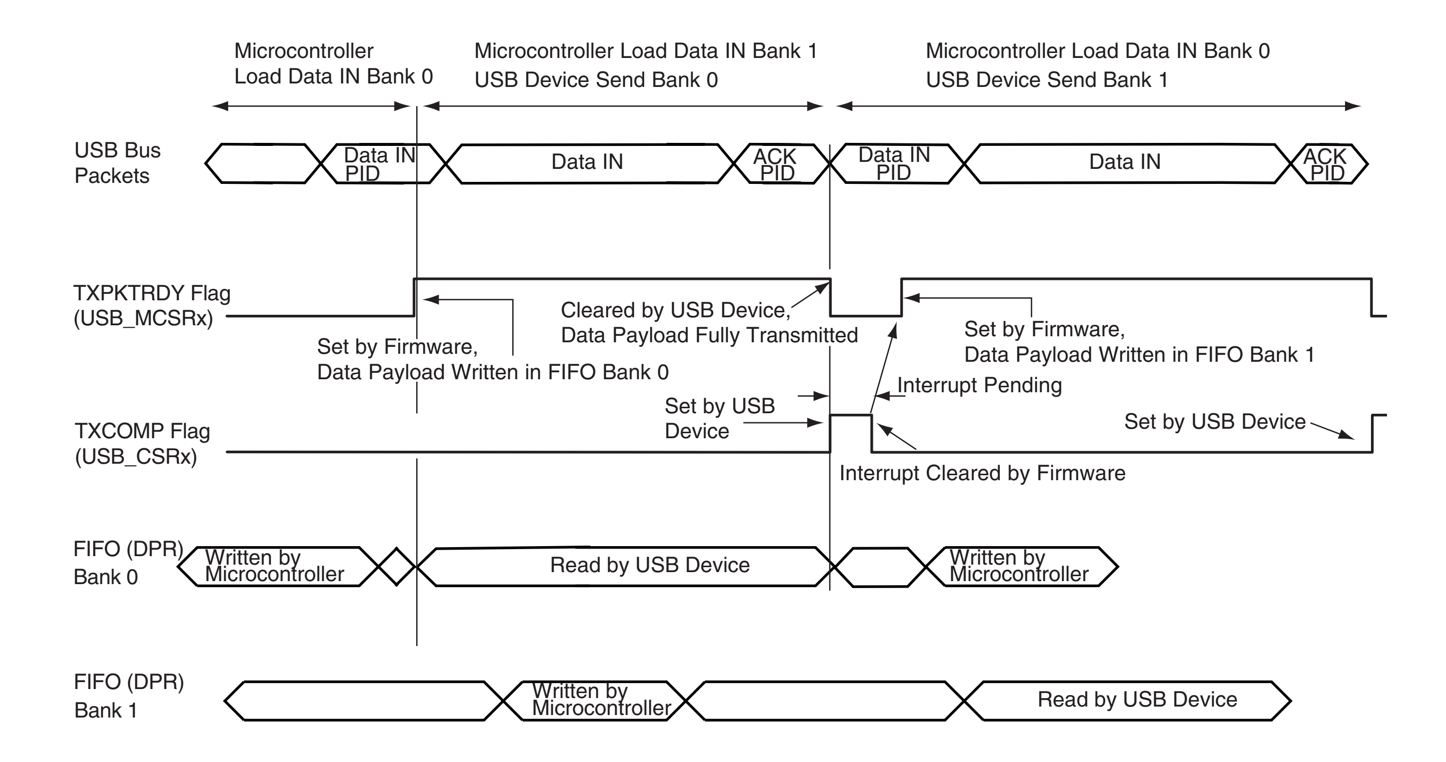
Figure 248. Ping-pong 端点数据入发送时的段交换



使用 ping-pong 端点时，数据入处理按以下步骤执行：

1. 微控制器通过轮询端点USB_CSRx寄存器TXPKTRDY是否清零来检查是否能在FIFO中写入。
2. 微控制器将要发送的第一个数据有效负载写入FIFO (段0) ,在端点USB_FDRx寄存器写入 0 或多字节值。
3. 微控制器通过设置端点 USB_CSRx 寄存器的 TXPKTRDY 位来通知 USB 外设 ,它已完成对 FIFO 段 0 的写入。
4. 不必等待 TXPKTRDY 清零，微控制器将第二个数据有效负载写入 FIFO (段 1)，在端点 USB_FDRx 寄存器写入 0 或多字节值。
5. 当 USB_CSRx 寄存器的 TXCOMP 置位， USB 器件通知微控制器已经释放首个段。TXCOMP 置位时将中断挂起。
6. 一旦微控制器收到首个段的TXCOMP时 ,它将端点USB_CSRx寄存器的TXPKTRDY拉高以通知 USB 器件准备发送第二个段数据。
7. 此步中，段 0 有效，微控制器开始准备发送第三个数据有效负载。

Figure 249. Ping-pong 端点的数据入发送



警告 : 这是软件关键路径 , 由于一旦第二个段满 , 驱动器须等待 TX_COMP 设置 TX_PKTRDY。若接收 TX_COMP 置位与 TX_PKTRDY 设置间延迟时间太长 , 某些数据入包可能无应答 , 减低带宽。

数据出处理

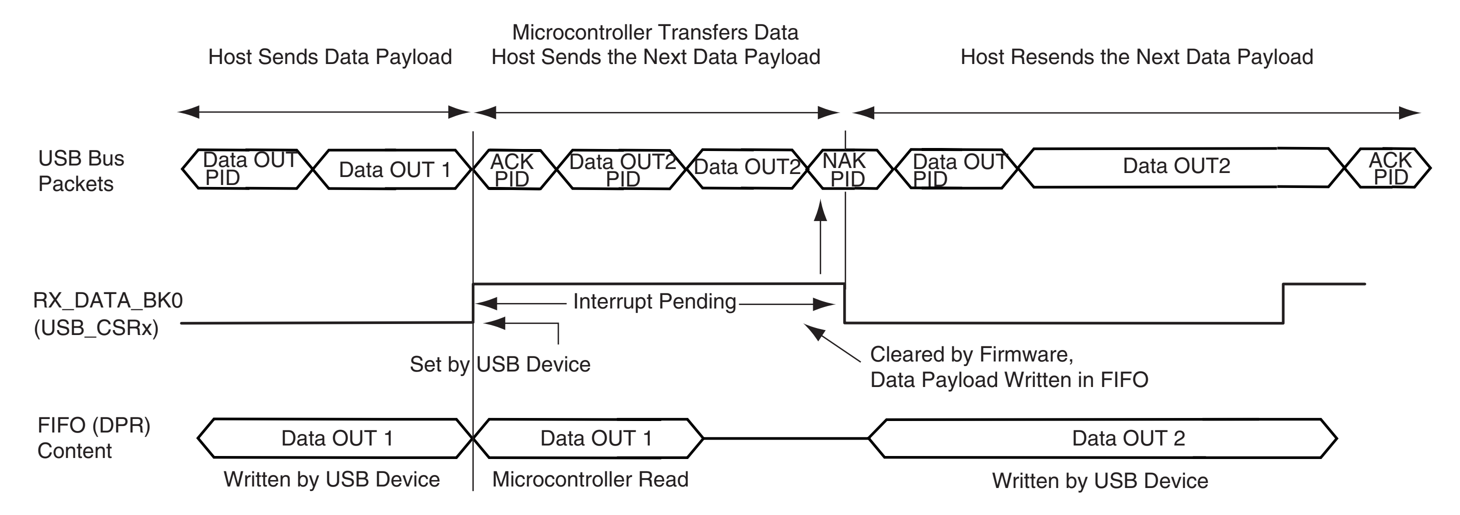
数据出处理用在控制、同步、批及中断传输中，并引导数据由主机向器件发送。同步发送中的数据出处理必须使用有 ping-pong 特性的端点中。

无 Ping-pong 特性的数据出处理

用无 ping-pong 端点执行数据出处理：

1. 主机产生数据出包。
2. USB 器件端点接收包。当与该端点相关的 FIFO 由微控制器使用时，NAK PID 信号返回主机。一旦 FIFO 有效， USB 器件将数据写入 FIFO 并自动给主机发送一个 ACK。
3. 通过轮询端点USB_CSRx寄存器的RX_DATA_BK0通知微控制器USB器件已接收到数据有效负载。当 RX_DATA_BK0 置位时，该端点中断挂起。
4. 读取端点 USB_CSRx 寄存器的 RXBYTECNT 得到 FIFO 中字节数目。
5. 微控制器接收端点存储器的数据到其存储器中。读端点 USB_FDRx 寄存器可得到接收数据。
6. 通过清零端点 USB_CSRx 寄存器的 RX_DATA_BK0，微控制器通知 USB 器件发送完成。
7. USB 器件可接收新数据出包。

Figure 250. 无 Ping-pong 端点数据出发送

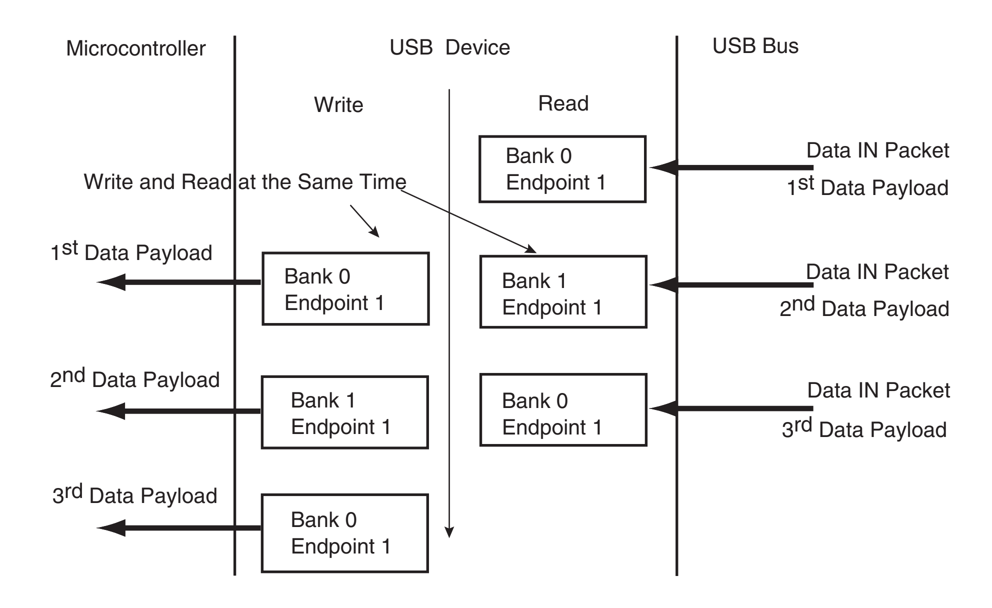


当 RX_DATA_BK0 标志置位，中断挂起。 RX_DATA_BK0 清零后， USB 器件、 FIFO 及微控制器存储器间发送不能进行。否则， USB 器件将接收下一个数据出发送并覆盖当前 FIFO 中的数据出包。

使用有 Ping-pong 特性的端点

同步发送时，必须使用有 ping-pong 特性端点。为保证带宽为常数，当 USB 器件接收当前数据有效负载时，微控制器必须读前一个主机发送的数据有效负载。因此要使用两个存储器段。一个给微控制器使用，另一个由 USB 器件锁定。

Figure 251. Ping-pong 端点数据出发送时的段交换

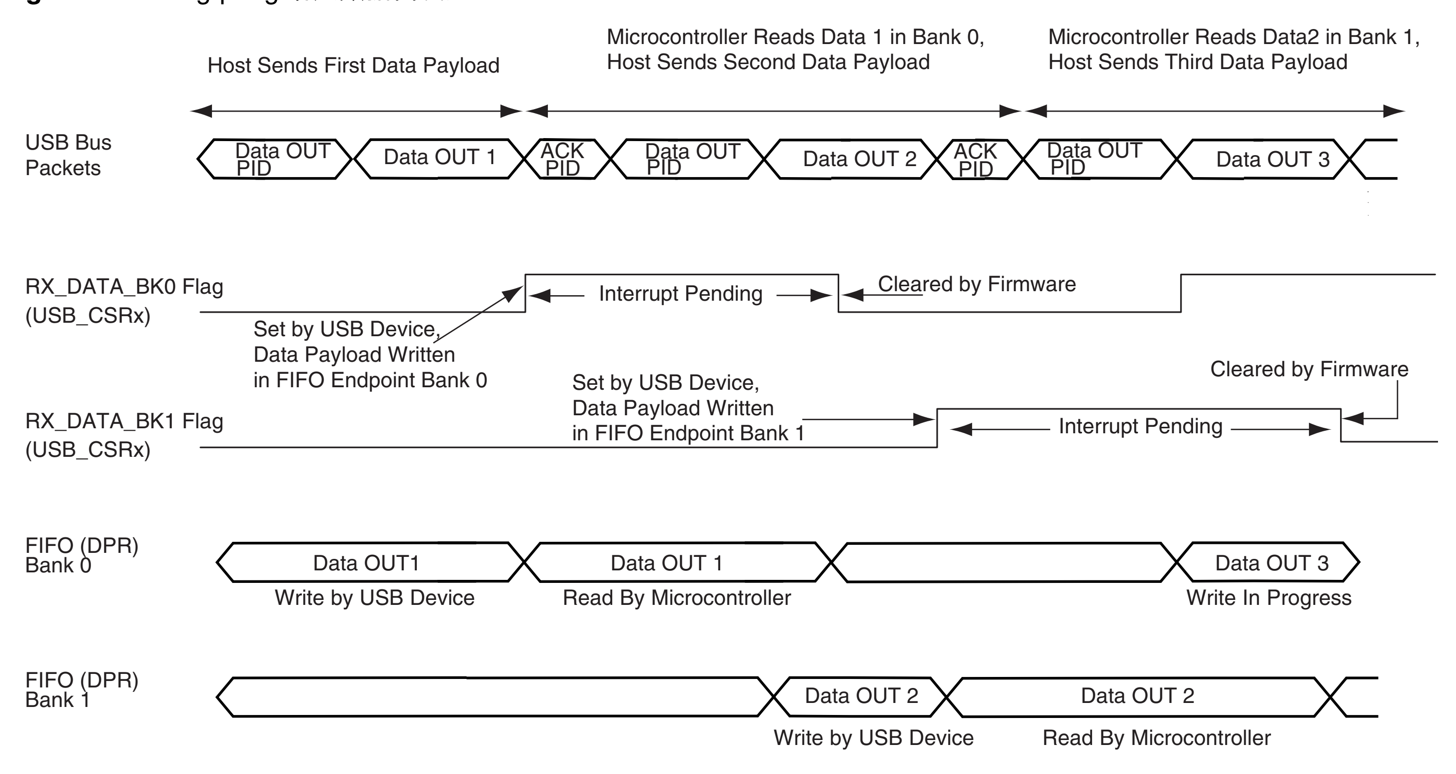


使用 ping-pong 端点时，数据出处理按以下步骤执行：

1. 主机产生数据出包。
2. USB 器件端点接收该包，写入端点 FIFO 段 0。
3. USB 器件向主机发送 ACK PID 包。主机可立即发送第二个数据出包。器件接收该包并复制到 FIFO 段 1。
4. 通过轮询端点USB_CSRx寄存器RX_DATA_BK0，通知微控制器USB器件已收到数据有效负载。 RX_DATA_BK0 置位时该端点中断挂起。
5. 读端点 USB_CSRx 寄存器的 RXBYTECNT 位可得到 FIFO 的可用字节数。
6. 微控制器将从端点存储器收到的数据发送打谱微控制器存储器。读端点 USB_FDRx 寄存器可得到接收到的数据。
7. 通过清零端点USB_CSRx寄存器的RX_DATA_BK0位，微控制器通知USB外设它已结束发送。
8. USB 外设器件可接收第三个数据出包，并复制到 FIFO 段 0。
9. 若已收到第二个数据出包，通过将端点USB_CSRx寄存器RX_DATA_BK1置位通知微控制器。当 RX_DATA_BK1 置位，端点中断挂起。
10. 微控制器将从端点存储器接收到的数据发送到微控制器存储器。读端点 USB_FDRx 寄存器可得到接收数据。
11. 通过清零端点USB_CSRx寄存器的RX_DATA_BK1位，微控制器通知USB器件它已完成发送。
12. USB 器件可接收第四个数据出包并复制到 FIFO 段 0。



Figure 252. Ping-pong 端点数据出发送



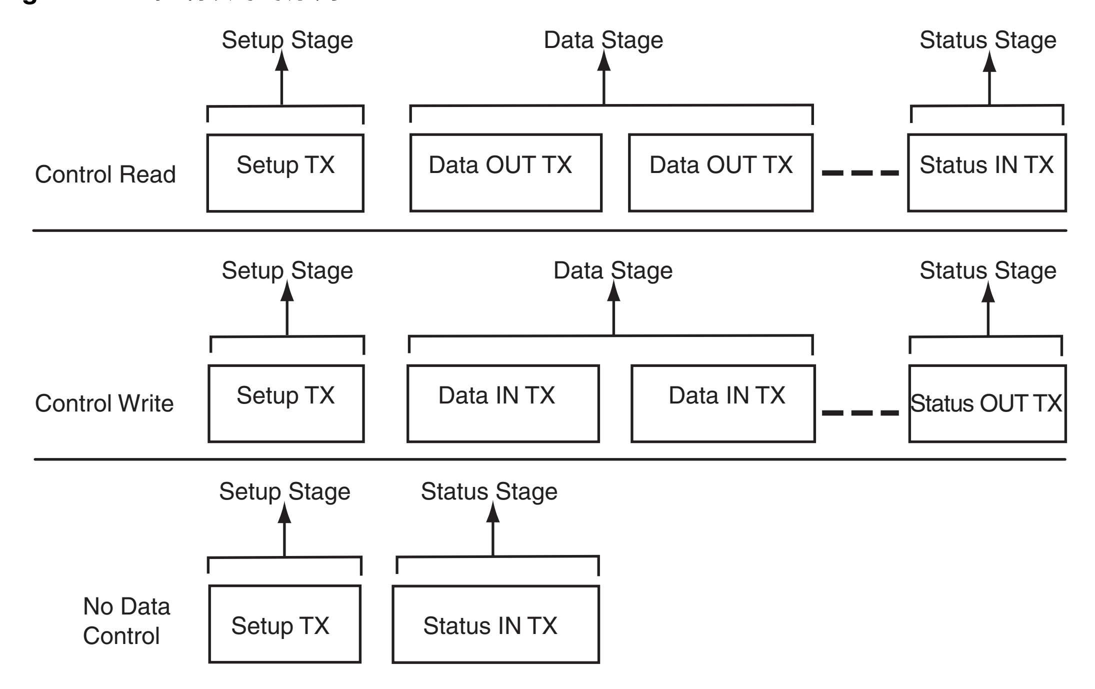
Note: 当 RX_DATA_BK0 或 RX_DATA_BK1 标志置位，中断挂起。

警告：当 RX_DATA_BK0 与 RX_DATA_BK1 均置位时，无法决定先对哪位清零。因此软件软件的内部计数器必须确保先清除 RX_DATA_BK0，然后再清除 RX_DATA_BK1。该状况可能在应用程序在别处忙且两段被 USB 主机填满时出现。一旦应用程序返回 USB 驱动器，两标志位置位。

状态处理

状态处理是控制发送中主机到器件处理的特殊类型。必须使用有 ping-pong 特性的端点执行控制发送。 USB 器件根据控制序列 (读或写) 发送或接收状态处理。

Figure 253. 控制读与写序列



Notes: 1. 状态入阶段 ,主机使用 DATA1 PID 等待来自器件的 0 长度包 (无数据的数据入处理)。详见*通用串行总线规范， Rev. 1.1* 第八章。
2. 状态出状态，主机发射 0 长度包到器件 (无数据的数据出处理)。

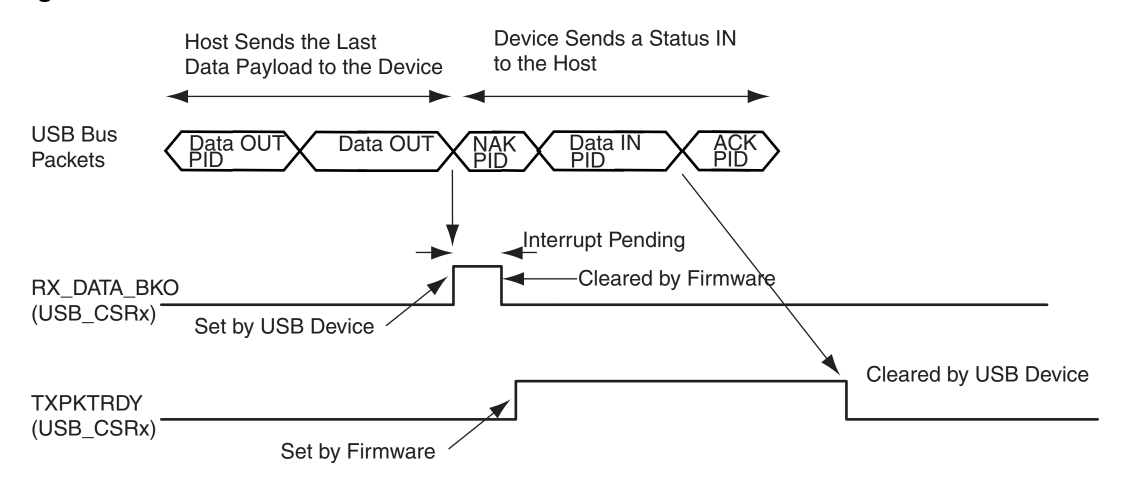


状态入发送

一旦处理控制请求，器件向主机返回状态。这是 0 长度数据入处理

1. 微控制器等待端点USB_CSRx寄存器TXPKTRDY位清零(该步骤中，必须对TXPKTRDY清零，因为前一个处理是设置处理或数据出处理)。
2. 不需向 USB_FDRx 寄存器写入任何值，微控制器设置 TXPKTRDY。 USB 器件使用 DATA1 PID 产生数据入包。
3. 该包由主机应答，并设置 USB_CSRx 寄存器的 TXPKTRDY 位。

Figure 254. 数据输出后状态入处理

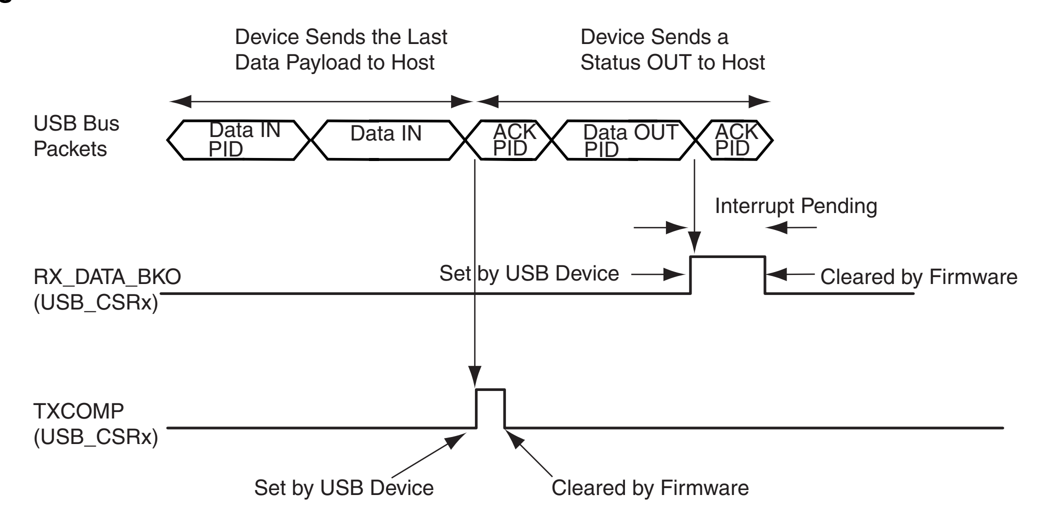


状态出发送

一旦处理控制请求且返回请求数据，主机通过发送 0 长度包应答。这是 0 长度数据出处理。

1. USB 器件接收 0 长度包。它设置 USB_CSRx 寄存器中的 RX_DATA_BK0 标志并应答 0 长度包。
2. 通过轮询USB_CSRx寄存器的RX_DATA_BK0位，通知微控制器USB器件已接收到主机发送的 0 长度包。当 RX_DATA_BK0 置位时，中断挂起。端点 USB_BCR 寄存器收到的字节数为 0。
3. 微控制器必须对 RX_DATA_BK0 清零。

Figure 255. 数据入后状态出处理



停止握手

停止握手用于下述两种场合之一 (更多内容见*通用串行总线规范， Rev 1.1*)。

- 当与端点相关的停止特性设置时使用功能停止 (更多内容见*通用串行总线规范，Rev 1.1* 第九章)。
- 中断当前请求，使用协议停止，仅适用于控制发送。

按以下步骤产生停止包：

1. 微控制器设置 USB_CSRx 寄存器的 FORCESTALL 标志。
2. 主机接收停止包。
3. 通过轮询 STALLSENT 设置来通知微控制器，器件已发送停止。当 STALLSENT 设置时端点中断挂起。微控制器必须对 STALLSENT 清零以清除中断。

当停止握手后收到设置处理时，必须对 STALLSENT 清零，以防止由于 STALLSENT 置位而引起的中断。

Figure 256. 停止握手 (数据入发送)

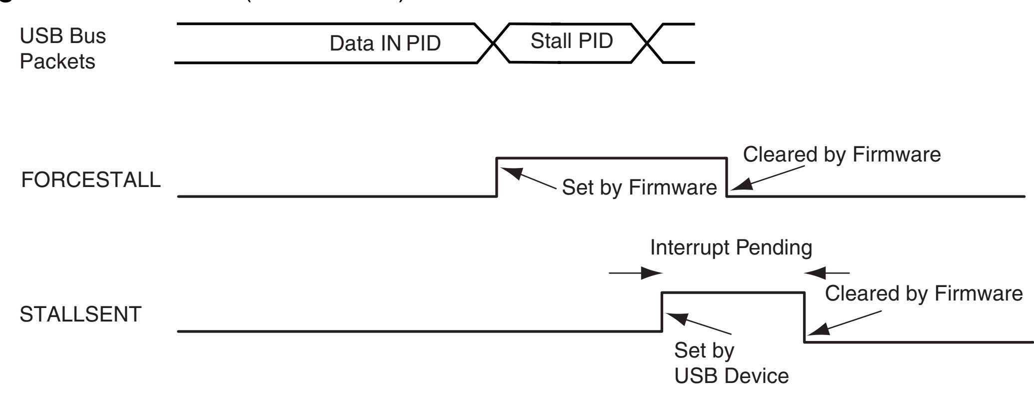
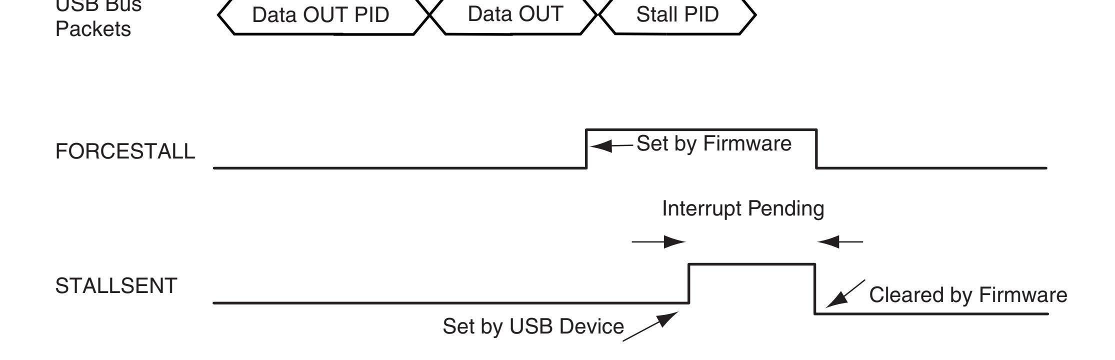


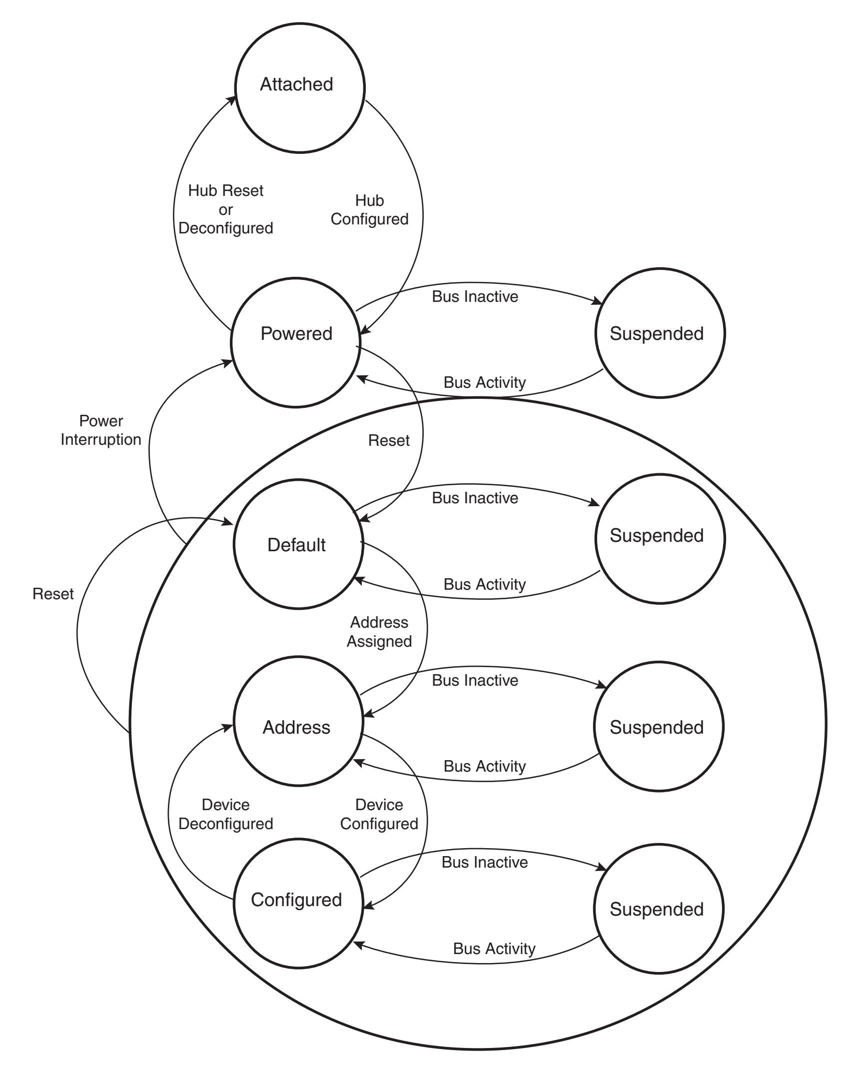
Figure 257. 停止握手 (数据出发送)



控制器件状态

USB 器件有几种可能状态，参见 *通用串行总线规范， Rev 1.1* 第九章。

Figure 258. USB 器件状态图



一个状态到另一个状态的转换取决于 USB 总线状态或通过默认端点 (端点 0) 的控制处理发送标准请求。

总线无效一周期后， UDP 器件进入挂起模式。强制接受 USB 主机挂起 / 恢复请求。挂起模式对总线供电应用要求非常严格；器件在 USB 总线上工作不超过 500 µA。

挂起模式时 , 主机通过发送恢复信号 (总线激活) 或 USB 器件向主机发送唤醒请求 以唤醒器件，例如通过移动 USB 鼠标来唤醒 PC。

唤醒特性并非对所有器件强制执行，并必须与主机协商。



由上电状态到默认状态

USB器件与USB主机连接后，它将等待一个总线结束复位。一旦在DP上检测到器件上拉，USB主机停止驱动复位状态。UDP_ISR 中未屏蔽标志 ENDBURST 置位并触发中断。UDP 软件使能默认端点，设置 UDP_CSR[0] 寄存器中 EPEDS 标志，并可通过在 UDP_IER 寄存器写 1 使能端点 0 中断（可选）。由控制发送开始列举。

由默认状态到地址状态

在设置地址标准器件请求后，USB 主机进入地址状态。此前，它实现控制发送的状态入处理，即一旦收到 UDP_CSR[0] 寄存器的 TXCOMP 位并将其清零后，UDP 器件设置其新地址。

为进入地址状态，驱动器软件设置 UDP_GLB_STATE 寄存器的 FADDEN 标志，设置其新地址，并设置 UDP_FADDR 寄存器 FEN 位。

由地址状态到配置状态

一旦收到有效设置配置标准请求并应答，器件使能相应当前配置的端点。通过设置 UDP_CSRx 寄存器上的 EPEDS 及 EPTYPE 域实现，并可在 UDP_IER 寄存器中使能相应中断（可选）。

挂起使能

当检测到挂起(USB总线上无总线活动)，设置UDP_ISR寄存器上RXSUSP信号。若UDP_IMR寄存器相应位置位则触发中断。

通过写 UDP_ICR 寄存器清除该标志。然后器件进入挂起模式。例如，微控制器切换到慢时钟，禁用 PLL 及主振荡器，并进入空闲模式。它也可以切断板上其它器件。

USB 器件外设时钟也可被切断。但收发器及 USB 外设不能切断，否则无法检测恢复。

接收主机恢复

挂起模式下，USB 收发器及 USB 外设必须上电以检测 RESUME。但由于 WAKEUP 为异步信号，可能无法对 USB 外设计时。

一旦在总线上检测到恢复，设置 UDP_ISR 寄存器的 WAKEUP 信号。若 UDP_IMR 寄存器相应位置位，可能产生一个中断。该中断可用来唤醒内核，使能 PLL 及主振荡器并配置时钟。设置 UDP_ICR 寄存器 WAKEUP 后，WAKEUP 位必须尽快清零。

发送外部恢复

外部恢复由主机处理，通过设置USB_GLB_STATE寄存器ESR位来使能。外部ext_resume_pin 上的异步事件产生 WAKEUP 中断。对于早期版本的 USP 外设，将立即在 USB 线上产生 K 状态。即 USB 器件必须可迅速对主机应答。新版本中，一旦与主机通信就绪，软件设置 UDP_GLB_STATE 寄存器的 RMWUPE 位。然后在总线上产生 K 状态。

通过设置 UDP_ICR 寄存器的 WAKEUP 位尽快将 WAKEUP 位清零。

USB 器件端口 (UDP) 用户接口

USB 器件端口 (UDP) 寄存器映射

偏移	寄存器	名称	访问类型	复位状态
0x000	帧数寄存器	USB_FRM_NUM	读	0x0000_0000
0x004	全局状态寄存器	USB_GLB_STAT	读 / 写	0x0000_0010
0x008	功能地址寄存器	USB_FADDR	读 / 写	0x0000_0100
0x00C	保留	–	–	–
0x010	中断使能寄存器	USB_IER	写	
0x014	中断禁用寄存器	USB_IDR	写	
0x018	中断屏蔽寄存器	USB_IMR	读	0x0000_1200
0x01C	中断状态寄存器	USB_ISR	读	0x0000_0000
0x020	中断清除寄存器	USB_ICR	写	
0x024	保留	–	–	–
0x028	复位端点寄存器	USB_RST_EP	读 / 写	
0x02C	保留	–	–	–
0x030	端点 0 控制与状态寄存器	USB _CSR0	读 / 写	0x0000_0000
0x034	端点 1 控制与状态寄存器	USB _CSR1	读 / 写	0x0000_0000
0x038	端点 2 控制与状态寄存器	USB _CSR2	读 / 写	0x0000_0000
0x03C	端点 3 控制与状态寄存器	USB _CSR3	读 / 写	0x0000_0000
0x040	端点 4 控制与状态寄存器	USB _CSR4	读 / 写	0x0000_0000
0x044	端点 5 控制与状态寄存器	USB _CSR5	读 / 写	0x0000_0000
0x048	端点 6 控制与状态寄存器	USB _CSR6	读 / 写	0x0000_0000
0x04C	端点 7 控制与状态寄存器	USB _CSR7	读 / 写	0x0000_0000
0x050	端点 0 FIFO 数据寄存器	USB_FDR0	读 / 写	0x0000_0000
0x054	端点 1 FIFO 数据寄存器	USB_FDR1	读 / 写	0x0000_0000
0x058	端点 2 FIFO 数据寄存器	USB_FDR2	读 / 写	0x0000_0000
0x05C	端点 3 FIFO 数据寄存器	USB_FDR3	读 / 写	0x0000_0000
0x060	端点 4 FIFO 数据寄存器	USB_FDR4	读 / 写	0x0000_0000
0x064	端点 5 FIFO 数据寄存器	USB_FDR5	读 / 写	0x0000_0000
0x068	端点 6 FIFO 数据寄存器	USB_FDR6	读 / 写	0x0000_0000
0x06C	端点 7 FIFO 数据寄存器	USB_FDR7	读 / 写	0x0000_0000
0x070	保留	–	–	–
0x074	保留	–	–	–



USB 帧数寄存器

寄存器名称 : USB_FRM_NUM

访问类型 : 只读

31	30	29	28	27	26	25	24
---	---	---	---	---	---	---	---

23	22	21	20	19	18	17	16
–	–	–	–	–	–	FRM_OK	FRM_ERR

15	14	13	12	11	10	9	8
–	–	–	–	–	FRM_NUM		

7	6	5	4	3	2	1	0
FRM_NUM							

- **FRM_NUM[10:0]: 包格式中定义的帧数目**

每有一帧，主机将这个 11 位值加一。每帧开始时更新此值。

在 SOF_EOP (帧开始，包结束) 时更新值。

- **FRM_ERR: 帧错误**

当接收到的 SOF 包中有错时，在 SOF_EOP 中的该位置位。

收到 SOF_PID 后该位复位。

- **FRM_OK: 帧正确**

当接收到的 SOF 包中无错时，在 SOF_EOP 中的该位置位。

收到 SOF_PID 后该位复位 (包标识)。

中断状态寄存器中，收到 SOF_PID 时更新 SOF 中断。该位置位不必等待 EOP。

Note: 在 8 位寄存器接口中， FRM_OK 为 FRM_NUM_H 的位 4， FRM_ERR 为 FRM_NUM_L 的位 3。

USB 全局状态寄存器

寄存器名称 : USB_GLB_STAT

访问类型 : 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	RMWUPE	RSMINPR	ESR	CONFG	FADDEN

该寄存器用于按照 *USB 串行总线规范 , Rev.1.1* 第九章的说明来获得并设置器件。

- **FADDEN: 功能地址使能**

读 :

0 = 器件不处于地址状态。

1 = 器件处于地址状态。

写 :

0 = 无效 , 只有复位能将器件带回默认状态。

1 = 将器件设置为地址状态。在成功设置地址请求后出现。在此之前 , 必须用设置地址参数将 USB_FADDR 寄存器初始化。在设置 FADDEN 前设置地址必须完成状态阶段 , 详见*通用串行总线规范 , Rev.1.1* 第九章的说明。

- **CONFG: 配置**

读 :

0 = 器件不处于配置状态。

1 = 器件处于配置状态。

写 :

0 = 设置器件到非配置状态。

1 = 设置器件到配置状态。

当处于地址状态且接收到成功设置配置请求时将器件置于配置状态 , 详见*通用串行总线规范 , Rev.1.1* 第九章的说明。

- **ESR: 使能发送恢复**

0 = 禁用远程唤醒序列。

1 = 能进行远程唤醒并使能引脚发送恢复。

- **RSMINPR: 已向主机发送恢复**

读 :

0 = 无效。

1 = 在远程唤醒中已向主机发送恢复。

- **RMWUPE: 远程唤醒使能**

0 = 收到任何主机包或 SOF 中断后必须清零。

1 = 若 ESR 使能 , 使能 USB 线上的 K 状态。



USB 功能地址寄存器

寄存器名称 : USB_FADDR

访问类型 : 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	FEN

7	6	5	4	3	2	1	0
–	FADD						

- **FADD[6:0]: 功能地址值**

一旦从主机接收到设置地址请求，并处于无数据控制序列的状态阶段，固件必须对功能地址值编程，详见*通用串行总线规范 , Rev.1.1* 的说明。上电或复位后，功能地址值设置为 0。

- **FEN: 功能使能**

读 :

0 = 功能端点禁用。

1 = 功能端点使能。

写 :

0 = 禁用功能端点。

1 = 默认值。

功能使能位 (FEN) 允许微控制器使能或禁用功能端点。从主机接收到复位后微控制器设置该位。一旦该位被设置， USB 器件被主机接受并可与主机传输数据包。

USB 中断使能寄存器

寄存器名称 : USB_IER

访问类型 : 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	WAKEUP	–	SOFINT	EXTRSM	RXRSM	RXSUSP

7	6	5	4	3	2	1	0
EP7INT	EP6INT	EP5INT	EP4INT	EP3INT	EP2INT	EP1INT	EP0INT

- **EP0INT: 使能端点 0 中断**
- **EP1INT: 使能端点 1 中断**
- **EP2INT: 使能端点 2 中断**
- **EP3INT: 使能端点 3 中断**
- **EP4INT: 使能端点 4 中断**
- **EP5INT: 使能端点 5 中断**
- **EP6INT: 使能端点 6 中断**
- **EP7INT: 使能端点 7 中断**

0 = 无效。

1 = 使能相应端点中断。

- **RXSUSP: 使能 USB 挂起中断**

0 = 无效。

1 = 使能 USB 挂起中断。

- **RXRSM: 使能 USB 恢复中断**

0 = 无效。

1 = 使能 USB 恢复中断。

- **EXTRSM: 使能外部恢复中断**

0 = 无效。

1 = 使能外部恢复中断。

- **SOFINT: 使能帧起始中断**

0 = 无效。

1 = 使能帧起始中断。

- **WAKEUP: 使能 USB 总线唤醒中断**

0 = 无效。

1 = 使能 USB 总线中断。



USB 中断禁用寄存器

寄存器名称 : USB_IDR

访问类型 : 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	WAKEUP	–	SOFINT	EXTRSM	RXRSM	RXSUSP

7	6	5	4	3	2	1	0
EP7INT	EP6INT	EP5INT	EP4INT	EP3INT	EP2INT	EP1INT	EP0INT

- **EP0INT: 禁用端点 0 中断**
- **EP1INT: 禁用端点 1 中断**
- **EP2INT: 禁用端点 2 中断**
- **EP3INT: 禁用端点 3 中断**
- **EP4INT: 禁用端点 4 中断**
- **EP5INT: 禁用端点 5 中断**
- **EP6INT: 禁用端点 6 中断**
- **EP7INT: 禁用端点 7 中断**

0 = 无效。

1 = 禁用相应端点中断。

- **RXSUSP: 禁用 USB 挂起中断**

0 = 无效。

1 = 禁用 USB 挂起中断。

- **RXRSM: 禁用 USB 恢复中断**

0 = 无效。

1 = 禁用 USB 恢复中断。

- **EXTRSM: 禁用外部恢复中断**

0 = 无效。

1 = 禁用外部恢复中断。

- **SOFINT: 禁用帧起始中断**

0 = 无效。

1 = 禁用帧起始中断。

- **WAKEUP: 禁用 USB 总线唤醒中断**

0 = 无效。

1 = 禁用 USB 总线中断。

USB 中断屏蔽寄存器

寄存器名称 : USB_IMR

访问类型 : 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	WAKEUP	–	SOFINT	EXTRSM	RXRSM	RXSUSP

7	6	5	4	3	2	1	0
EP7INT	EP6INT	EP5INT	EP4INT	EP3INT	EP2INT	EP1INT	EP0INT

- **EP0INT: 屏蔽端点 0 中断**
- **EP1INT: 屏蔽端点 1 中断**
- **EP2INT: 屏蔽端点 2 中断**
- **EP3INT: 屏蔽端点 3 中断**
- **EP4INT: 屏蔽端点 4 中断**
- **EP5INT: 屏蔽端点 5 中断**
- **EP6INT: 屏蔽端点 6 中断**
- **EP7INT: 屏蔽端点 7 中断**

0 = 禁用相应端点中断。

1 = 使能相应端点中断。

- **RXSUSP: 屏蔽 USB 挂起中断**

0 = 禁用 USB 挂起中断。

1 = 使能 USB 挂起中断。

- **RXRSM: 屏蔽 USB 恢复中断**

0 = 禁用 USB 恢复中断。

1 = 使能 USB 恢复中断。

- **EXTRSM: 屏蔽外部恢复中断**

0 = 禁用外部恢复中断。

1 = 使能外部恢复中断。

- **SOFINT: 屏蔽帧起始中断**

0 = 禁用帧起始中断。

1 = 使能帧起始中断。

- **WAKEUP: USB 总线唤醒中断**

0 = 禁用 USB 总线唤醒中断。

1 = 使能 USB 总线唤醒中断。

Note: 当 USB 块处于挂起模式时，应用程序可能关闭 USB 逻辑。此时，所有 USB 主机恢复请求必须记录，因此，USB_IMR 寄存器的 RXRSM 位复位值必须使能。



USB 中断状态寄存器

寄存器名称： USB_ISR

访问类型： 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	WAKEUP	ENDBUSRES	SOFINT	EXTRSM	RXRSM	RXSUSP

7	6	5	4	3	2	1	0
EP7INT	EP6INT	EP5INT	EP4INT	EP3INT	EP2INT	EP1INT	EP0INT

- **EP0INT: 中断 0 中断状态**

0 = 无端点 0 中断挂起。

1 = 端点 0 中断已发生。

几个信号可产生该中断，通过读 USB_CSR0 可得：

RXSETUP 置为 1

RX_DATA_BK0 置为 1

RX_DATA_BK1 置为 1

TXCOMP 置为 1

STALLSENT 置为 1

EP0INT 为 粘着位。写相应 USB_CSR0 位对 EP0INT 清零前中断有效。

- **EP1INT: 端点 1 中断状态**

0 = 无端点 1 中断挂起。

1 = 端点 1 中断已发生。

几个信号可产生该中断，通过读 USB_CSR1 可得：

RXSETUP 置为 1

RX_DATA_BK0 置为 1

RX_DATA_BK1 置为 1

TXCOMP 置为 1

STALLSENT 置为 1

EP1INT 为 粘着位。写相应 USB_CSR1 位对 EP1INT 清零前中断有效。

- **EP2INT: 端点 2 中断状态**

0 = 无端点 2 中断挂起。

1 = 端点 2 中断已发生。

几个信号可产生该中断，通过读 USB_CSR2 可得：

RXSETUP 置为 1

RX_DATA_BK0 置为 1

RX_DATA_BK1 置为 1

TXCOMP 置为 1

STALLSENT 置为 1

EP2INT 为 粘着位。写相应 USB_CSR2 位对 EP2INT 清零前中断有效。

- **EP3INT: 端点 3 中断状态**

0 = 无端点 3 中断挂起。

1 = 端点 3 中断已发生。

几个信号可产生该中断，通过读 USB_CSR3 可得：

RXSETUP 置为 1

RX_DATA_BK0 置为 1

RX_DATA_BK1 置为 1

TXCOMP 置为 1

STALLSENT 置为 1

EP3INT 为 粘着位。写相应 USB_CSR3 位对 EP3INT 清零前中断有效。

- **EP4INT: 端点 4 中断状态**

0 = 无端点 4 中断挂起。

1 = 端点 4 中断已发生。

几个信号可产生该中断，通过读 USB_CSR4 可得：

RXSETUP 置为 1

RX_DATA_BK0 置为 1

RX_DATA_BK1 置为 1

TXCOMP 置为 1

STALLSENT 置为 1

EP4INT 为 粘着位。写相应 USB_CSR4 位对 EP4INT 清零前中断有效。

- **EP5INT: 端点 5 中断状态**

0 = 无端点 5 中断挂起。

1 = 端点 5 中断已发生。

几个信号可产生该中断，通过读 USB_CSR5 可得：

RXSETUP 置为 1

RX_DATA_BK0 置为 1

RX_DATA_BK1 置为 1

TXCOMP 置为 1

STALLSENT 置为 1

EP5INT 为 粘着位。写相应 USB_CSR5 位对 EP5INT 清零前中断有效。

- **EP6INT: 端点 6 中断状态**

0 = 无端点 6 中断挂起。

1 = 端点 6 中断已发生。

几个信号可产生该中断，通过读 USB_CSR6 可得：

RXSETUP 置为 1

RX_DATA_BK0 置为 1

RX_DATA_BK1 置为 1

TXCOMP 置为 1

STALLSENT 置为 1

EP6INT 为 粘着位。写相应 USB_CSR6 位对 EP6INT 清零前中断有效。



- **EP7INT: 端点 7 中断状态**

0 = 无端点 7 中断挂起。

1 = 端点 7 中断已发生。

几个信号可产生该中断，通过读 USB_CSR7 可得：

RXSETUP 置为 1

RX_DATA_BK0 置为 1

RX_DATA_BK1 置为 1

TXCOMP 置为 1

STALLSENT 置为 1

EP7INT 为 粘着位。写相应 USB_CSR7 位对 EP7INT 清零前中断有效。

- **RXSUSP: USB 挂起中断状态**

0 = 无 USB 挂起中断等待。

1 = USB 挂起中断已发生。

当 USB 器件检测到 3ms 无工作时设置该位。 USB 器件进入挂起模式。

- **RXRSM: USB 恢复中断状态**

0 = 无 USB 恢复中断等待。

1 =USB 恢复中断已发生。

当 USB 器件端口检测到 USB 恢复信号时设置该位。

- **EXTRSM: 外部恢复中断状态**

0 = 无外部恢复中断等待。

1 = 外部恢复中断已发生。

挂起模式下，在 send_resume 检测到一个异步上升沿时发生中断。

若 RMWUPE = 1， USB 总线上发送一个恢复状态。

- **SOFINT: 帧开始中断状态**

0 = 无帧开始中断等待。

1 = 帧开始中断已发生。

每次检测到 SOF 时发生中断。它可作为同步信号使用同步端点。

- **ENDBUSRES: 总线结束复位中断状态**

0 = 无总线结束复位中断等待。

1 = 总线结束复位中断已发生。

USB 复位序列结束后发生中断。 USB 器件必须在端点 0 准备接收请求。主机开始列举，然后开始进行配置。

- **WAKEUP: USB 恢复中断状态**

0 = 无唤醒中断等待。

1 = 上次清零后出现唤醒中断 (USB 主机发送 RESUME 或 RESET)。

USB 中断清除寄存器

寄存器名称 : USB_ICR

访问类型 : 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	WAKEUP	ENDBURST	SOFINT	EXTRSM	RXRSM	RXSUSP

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

- **RXSUSP: 清除 USB 挂起中断**

0 = 无效。

1 = 清除 USB 挂起中断。

- **RXRSM: 清除 USB 恢复中断**

0 = 无效。

1 = 清除 USB 恢复中断。

- **EXTRSM: 清除外部恢复中断**

0 = 无效。

1 = 清除外部恢复中断。

- **SOFINT: 清除帧开始中断**

0 = 无效。

1 = 清除帧开始中断。

- **ENDBURST: 清除总线复位结束中断**

0 = 无效。

1 = 清除总线复位结束中断。

- **WAKEUP: 清除唤醒中断**

0 = 无效。

1 = 清除唤醒中断。



USB 复位端点寄存器

寄存器名称 : USB_RST_EP

访问类型 : 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
EP7	EP6	EP5	EP4	EP3	EP2	EP1	EP0

- **EP0: 复位端点 0**
- **EP1: 复位端点 1**
- **EP2: 复位端点 2**
- **EP3: 复位端点 3**
- **EP4: 复位端点 4**
- **EP5: 复位端点 5**
- **EP6: 复位端点 6**
- **EP7: 复位端点 7**

该标志用于复位与端点相关的 FIFO 及 UDP_CSRx 寄存器中的 RXBYTECOUNT 位。它还复位数据切换到 DATA0。它在删除 BULK 端点 HALT 状态后有用。参见 *USB 串行总线规范 , Rev.1.1* 的 5.8.5 节。

警告 : 复位结束时该标志必须清除。不清除 USB_CSRx 标志。

0 = 无复位。

1 = 强制将相应端点 FIF0 指向 0 , 因此 USB_CSRx 寄存器 RXBYTECNT 域为 0。

USB 端点控制与状态寄存器

寄存器名称 : USB_CSRx [x = 0. 7]

访问类型 : 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	RXBYTECNT		

23	22	21	20	19	18	17	16
RXBYTECNT							

15	14	13	12	11	10	9	8
EPEDS	–	–	–	DTGLE	EPTYPE		

7	6	5	4	3	2	1	0
DIR	RX_DATA_ BK1	FORCE STALL	TXPKTRDY	STALLSENT ISOERROR	RXSETUP	RX_DATA_ BK0	TXCOMP

- **TXCOMP: 产生一个有前数据写入 DPR 的入包**

该标志设置为 1 时产生中断。

写 (由固件清零) ;

0 = 清除标志 , 清除中断。

1 = 无效。

读 (由 USB 外设设置) :

0 = 主机未应答数据入处理。

1 = 数据入处理完成 , 主机应答。

数据入处理设置 TXPKTRDY 后 , 器件固件等待 TXCOMP 以确保主机已应答处理。

- **RX_DATA_BK0: 接收数据段 0**

该标志设置为 1 时产生中断。

写 (由固件清零) ;

0 = 通知 USB 外设器件 , FIFO 段 0 数据已被读取。

1 = 无效。

读 (由 USB 外设设置) :

0 = FIFO 段 0 未收到数据包。

1 = 已收到数据包 , 并已存入 FIFO 段 0。

当器件固件已轮询该位或由该信号中断 , 必须将数据由 FIFO 送到微控制器存储器。 RXBYTCENT 域中收到的字节数有效。通过 USB_FDRx 寄存器读 FIFO 段 0 值。一旦发送完成 , 器件固件必须通过清除 RX_DATA_BK0 将 USB 外设器件段 0 释放。

- **RXSETUP: 向主机发送 STALL(控制端点)**

该标志设置为 1 时产生中断。

读 :

0 = 无有效设置包。

1 = 主机已发送设置包并在 FIFO 中有效。

写 :

0 = 器件固件通知 USB 外设器件 , 它已读取 FIFO 中设置数据。

1 = 无效。

该标志用来通知 USB 器件固件 , 主机已收到有效设置数据包并由 USB 器件成功接收。 USB 器件固件可通过读 USB_FDRx 寄存器将 FIFO 中设置数据发送到微控制器处理器。一旦发送完成 , RXSETUP 必须由器件固件清除。



当 RXSETUP 置位时，不接收随后的数据出数据。

- **STALLSENT: 发送停止 (控制、批中断端点) / ISOERROR (同步端点)**

该标志设置为 1 时产生中断。

STALLSENT：结束停止握手。

读：

0 = 主机未应答 STALL。

1 = 主机应答 STALL。

写：

0 = 复位 STALLSENT 标志，清除中断。

1 = 无效。

强制器件固件清除该标志，否则中断保持。

关于 STALL 握手，参见*通用串行总线规范， Rev. 1.1* 8.4.5 及 9.4.5 节。

ISOERROR：同步发送中检测到 CRC 错误。

读：

0 = 之前的同步传输中无错误。

1 = 检测到 CRC 错误，FIFO 中可用数据被破坏。

写：

0 = ISOERROR 标志复位，清除中断。

1 = 无效。

- **TXPKTRDY: 发送包就绪**

该标志由 USB 器件清除。

该标志由 USB 器件固件置位。

读：

0 = 数据值可写入 FIFO。

1 = 数据值不能写入 FIFO。

写：

0 = 无效。

1 = 新数据有效负载通过固件写入 FIFO 并即将发送。

该标志用来产生数据入处理 (器件到主机)。器件固件检查它能否在 FIFO 中写入数据有效负载 ,检查 TXPKTRDY 是否清零。通过写 USB_FDRx 寄存器实现数据到 FIFO 的发送。一旦数据有效否则发送到 FIFO，固件通知 USB 器件将 TXPKTRDY 设置为 1 。 USB 总线处理开始。一旦主机收到数据有效负载， TXCOMP 置位。

- **FORCESTALL: 强制停止 (用于控制、批及同步端点)**

只写

0 = 无效。

1 = 向主机发送 STALL。

关于 STALL 握手，参见*通用串行总线规范， Rev. 1.1* 8.4.5 及 9.4.5 节。

控制端点：数据筹备与状态筹备过程中，表明微控制器不能完成请求。

批与中断端点：通知主机端点已停止。

主机应答 STALL， STALLSENT 标志通知器件固件。

- **RX_DATA_BK1: 接收数据段 1 (只用于有 ping-pong 特性的端点)**

该标志设置为 1 时产生中断。

写 (由固件清零)；

0 = 通知 USB 器件，FIFO 段 1 数据已被读取。

1 = 无效。

读 (由 USB 外设设置) :

0 = FIFO 段 1 未收到数据包。

1 = 已收到数据包，并已存入 FIFO 段 1。

当器件固件已轮询该位或由该信号中断，必须将数据由 FIFO 送到微控制器存储器。 RXBYTCENT 域中收到的字节数有效。通过 USB_FDRx 寄存器读 FIFO 段 1 值。一旦发送完成 , 器件固件必须通过清除 RX_DATA_BK1 将 USB 外设器件段 1 释放。

- **DIR: 发送方向 (仅对控制端点有效)**

读 / 写

0 = 在控制数据筹备时允许数据出处理。

1 = 控制数据筹备时使能数据入处理。

关于控制数据筹备，参见*通用串行总线规范， Rev. 1.1* 8.5.3 节。

在设置筹备结束清除 USB_CSRx/RXSETUP 前该位必须置位。根据设置数据包发送请求，数据筹备为器件到主机 (DIR = 1) 或主机到器件 (DIR = 0) 数据发送。状态筹备时不需检验该位反向。

- **EPTYPE[2:0]: 端点类型**

读 / 写

000	控制
001	同步出
101	同步入
010	批出
110	批入
011	中断出
111	中断入

- **DTGLE: 数据切换**

只读

0 = 识别 DATA0 包。

1 = 识别 DATA1 包。

更多 DATA0、 DATA1 包定义见*通用串行总线规范， Rev. 1.1* 第八章。

- **EPEDS: 端点使能禁用**

读 :

0 = 端点禁用。

1 = 端点使能。

写 :

0 = 禁用端点。

1 = 使能端点。

- **RXBYTECNT[10:0]: FIFO 中可用字节数**

只读

当主机向器件发送数据包时， USB 器件将数据存入 FIFO 并通知微控制器。微控制器可通过读取 USB_FDRx 寄存器的 RXBYTECENT 字节载入 FIFO 中数据。



USB FIFO 数据寄存器

寄存器名称 : USB_FDRx [x = 0. 7]

访问类型 : 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
FIFO_DATA							

- **FIFO_DATA[7:0]: FIFO 数据值**

微控制器可通过该寄存器将数据推入或弹出 FIFO。

相应 USB_CSRx 寄存器的 RXBYTECNT 中是由 FIFO 中读取的字节数 (主机发送)。

能写入的最大字节数由标准端点描述符的最大包尺寸确定。它不能大于相关端点物理存储器大小，详见*通用串行总线规范，Rev. 1.1*。

USB 主机端口 (UHP)

概述

USB主机端口在主机应用中与USB连接。它处理开HCI协议(开主机控制器接口)及USB v2.0全速与低速协议。它还给 ASB 提供简单的读 / 写协议。

USB 主机端口集成一个根集线器，并在下游端口集成收发器。它提供几个半双工高速串行通信端口，速率为 12 Mbit/s。最多可连接 127 个 USB 器件 (打印机、照相机、鼠标、键盘、硬盘等)，而 USB 集线器可使用 " 分层星型 " 布局与 USB 主机连接。

USB主机端口控制器与开HCI规范完全兼容。标准OHCI USB堆栈驱动器可以相同的方式轻松进入 ATMEL 已有的无硬件规范的驱动结构中。

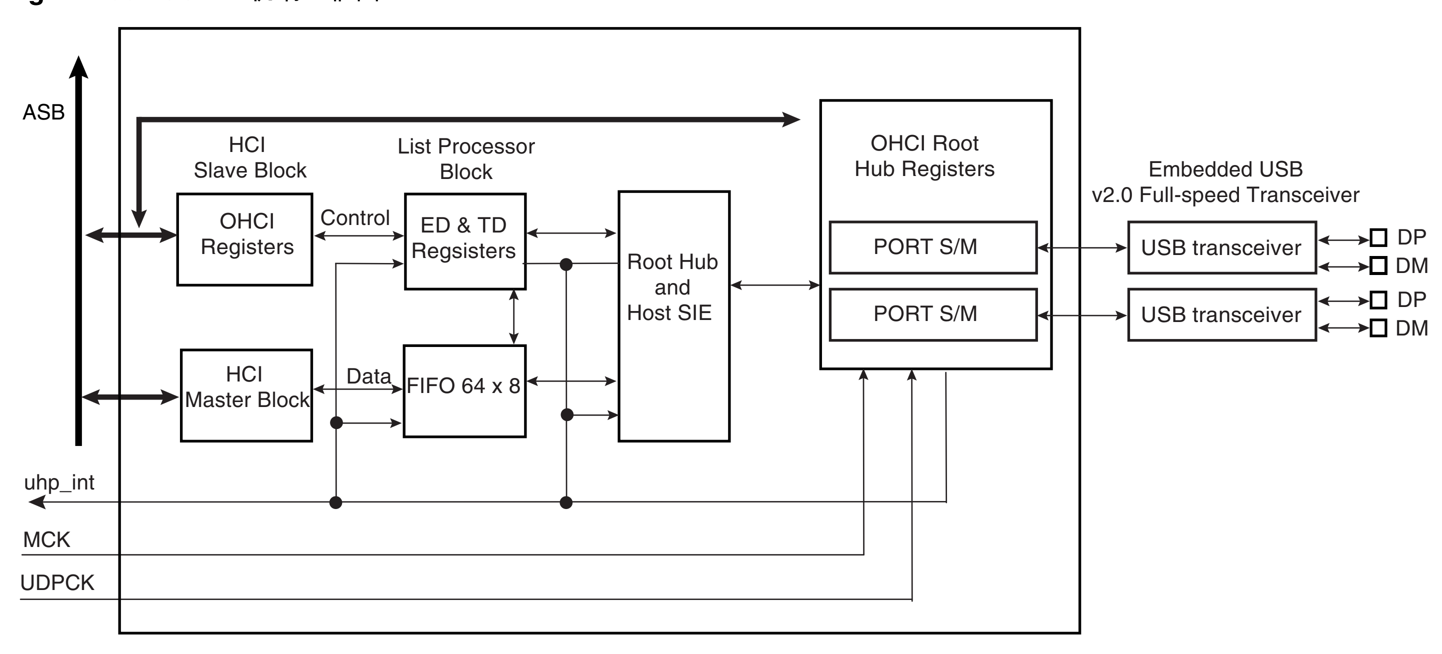
这意味着标准分类驱动是自动检测并可在用户应用中使用。例如，集成 HID (人机接口设备) 类驱动器对所有 USB 键盘与鼠标提供即插即用功能。

USB 主机端口主要特性如下：

- 与开 HCI Rev 1.0 规范兼容
- 与 USB V2.0 全速及低速规范兼容
- 支持低速 1.5 Mbps 与全速 12 Mbps 的 USB 设备
- 有两个下游 USB 端口的根集线器
- 内置 USB 收发器 (收发器数目由产品决定)
- 支持电源管理
- 在 ASB 总线上作为主机工作

方框图

Figure 259. USB 主机端口框图



通过 ASB 总线接口访问 USB 主机工作寄存器。开 HCI 主机控制器使用 ASB 总线初始化主机 DMA 传输：

- 取得端点描述符与传输描述符
- 从系统存储器访问端点数据



- 访问 HC 通信域
- 写状态并退出传输描述符

USB主机的DMA可访问所有的ASB存储器映射。因此不必在USB主机中定义专用的物理存储器空间。

USB根集线器集成在USB主机上 。几个USB下游端口有效。下游端口数目可通过软件驱动器对根集线器工作寄存器的读取来确定。由 USB 主机端口逻辑自动检测器件连接。

警告：DP 上必须连接下拉电阻。否则 USB 主机将始终在该端口检测器件连接。

USB 物理收发器集成在产品中并由根集线器端口驱动。

USB 主机控制器能激活端口过电流保护。 Atmel 标准产品没有专用的外部过电流保护垫。

附属产品

I/O 线

PIO 控制器不能控制 DP 与 DM。内置的 USB 物理收发器由 USB 主机控制器控制。

电源管理

USB 主机控制器需要 48 MHz 时钟。该时钟必须由精度为 ± 0.25% 的 PLL 产生。

因此， USB 器件外设从电源管理控制器 (PMC) 收到两个时钟：用来驱动外设用户接口的主机时钟 (MCK 域) 及用来与总线 USB 序号连接的 UHPCLK 48 MHz 时钟 (恢复 12 MHz 域)。

中断

USB 主机接口有一条与高级中断控制器 (AIC) 连接的中断线。

处理 USB 主机中断请求需要在配置 UHP 前对 AIC 编程。

功能说明

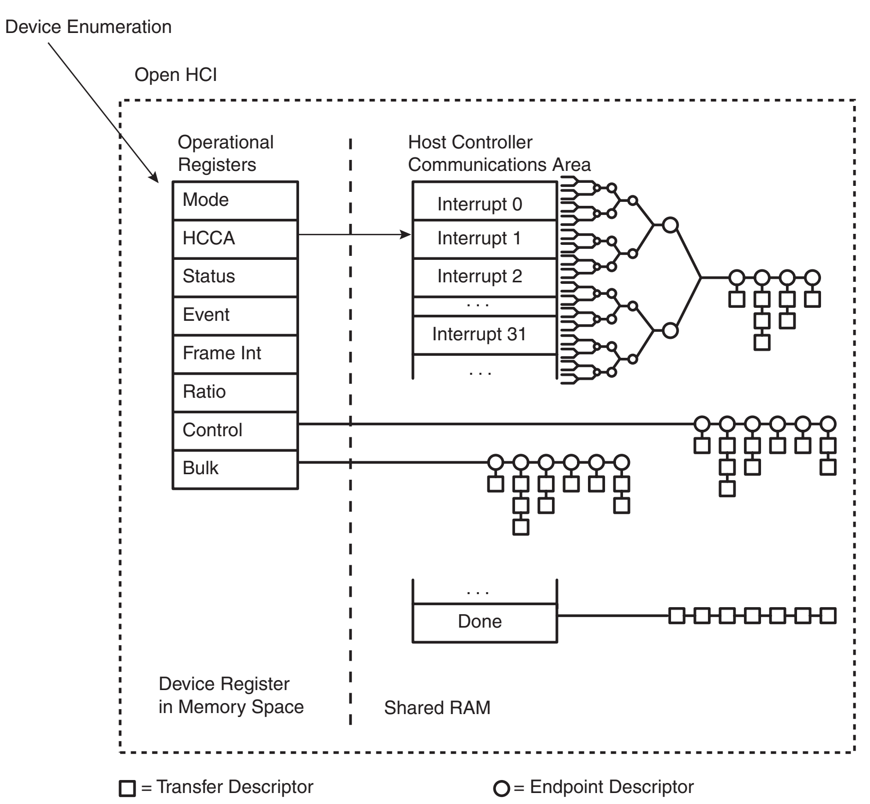
请参见 USB 版本 1.0.a 的开主机控制器接口规范。

主机控制器接口

主机控制器与主机控制器驱动器间有两个信道。第一个通道使用一套位于 USB 主机控制器上的工作寄存器。该通道的通信目标是主机控制器。工作寄存器包括控制、状态及列表指针寄存器。它们映射在 ASB 存储器映射区。在工作寄存器中有一个位于处理器地址空间的指针，称为主机控制器通信域 (HCCA)。 HCCA 为第二条信道。主机控制器为主机在该信道上的所有通信。HCCA包括对中断端点描述符列表的头指针、已排队及与帧启动处理相关的状态信息的头指针。

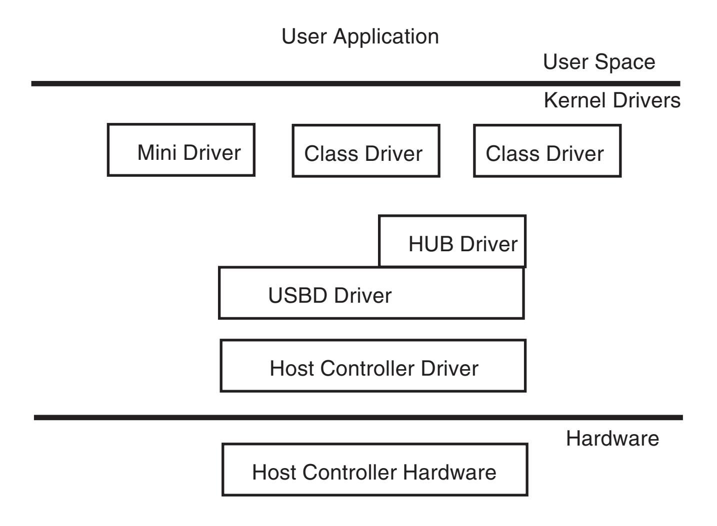
通过接口通信的基本模块为：端点描述符 (ED， 4 个双字) 与传输描述符 (TD， 4 或 8 个双字)。主机控制器对系统中每个端点分配一个端点描述符。传输队列描述符与具体端点的端点描述符连接。

Figure 260. USB 主机通信通道



主机控制驱动器

Figure 261. USB 主机控制驱动器

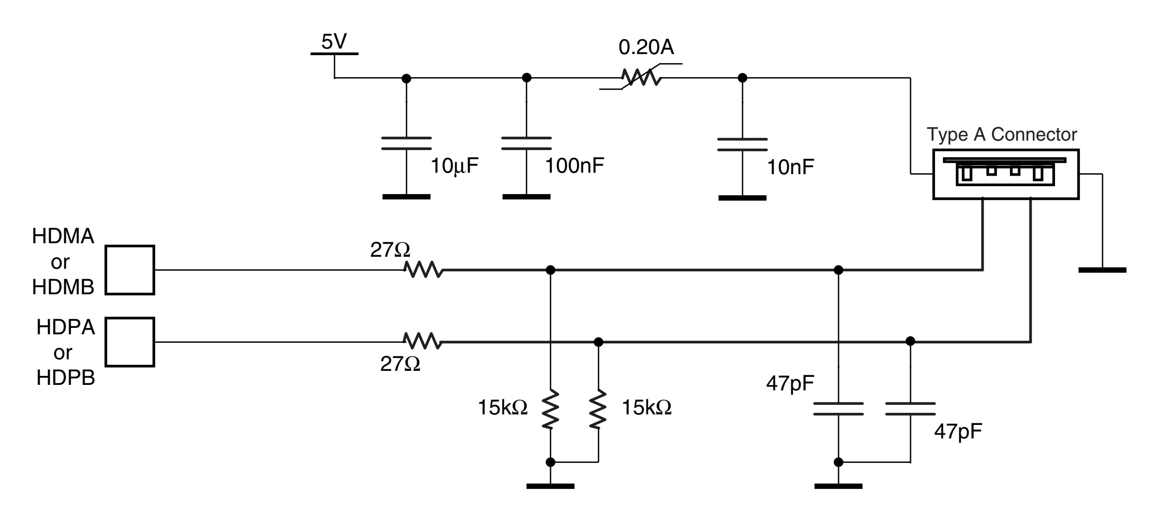


USB 处理通过以下几层进行：

- 主机控制器硬件与串行引擎：在总线上发送与接收 USB 数据。
- 主机控制器驱动器：驱动主机控制器硬件并处理 USB 协议。
- USB 总线驱动器与集线器驱动器：处理 USB 命令并列举。提供硬件独立接口。
- 小驱动器：处理器件具体命令。
- 类驱动器：处理标准器件。作为一类器件的通用驱动器，例如 HID 驱动器。

典型连接

Figure 262. UHP 器件控制器连接示意图



由于 USB 主机端口逻辑自动检测器件连接，因此 DP 与 DM 上必须有下拉电阻。否则 USB 主机将一直检测该端口器件连接。

以太网 MAC (EMAC)

概述

以太网 MAC 是 OSI 参考模型物理层 (PHY) 与逻辑链路层 (LLC) 间 MAC 子层的硬件工具。它使用以太网 IEEE 802.3u 数据帧格式控制在主机与 PHY 层间的数据交换。以太网 MAC 包括所需逻辑与 DMA 管理的发送与接收 FIFO。此外，它通过与 MDIO/MDC 引脚连接来对 PHY 层进行管理。

以太网 MAC 根据引脚输出配置不同，可使用独立媒体接口 (MII) 或简化独立媒体接口 (RMII) 来传输数据。

简化接口的目的是减少接线器引脚数，以连接多 PHY 接口。 RMII 模式特性如下：

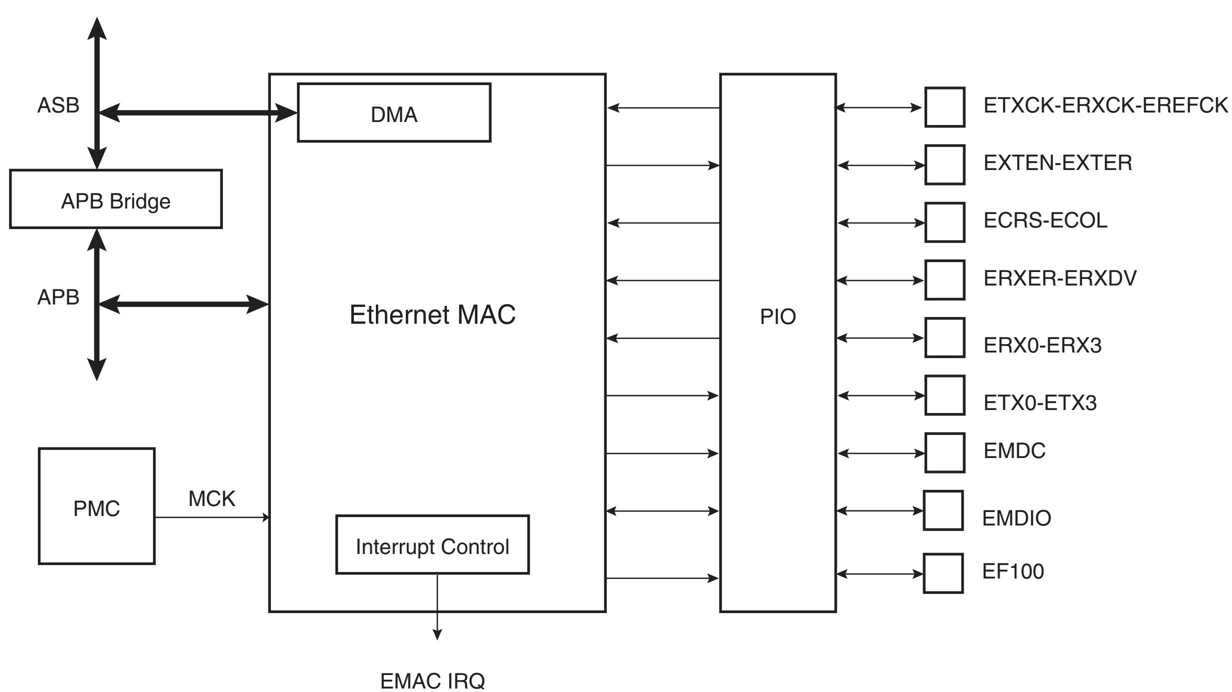
- 单时钟频率为 50 MHz
- 缩减所需控制引脚
- 通过加倍时钟频率缩减数据路径到 2 位宽
- 10 Mbits/s 与 100 Mbits/s 的数据传输能力

EMAC 主要特性如下：

- 与 IEEE 标准 802.3 兼容
- 10 或 100 Mbits 每秒的数据吞吐能力
- 全双工或半双工操作
- 对物理层的 MII 或 RMII 接口
- 寄存器与地址、状态及控制寄存器连接
- DMA 接口
- 信号接收与发送完成中断产生
- 28 字节发送与 28 字节接收 FIFO
- 发送帧的自动焊盘与 CRC 产生
- 地址检查逻辑辨识四个 48 位地址
- 支持混合模式，将所有有效帧复制到存储器中
- 支持物理层管理，通过 MDIO 接口控制报警与更新时间 / 日历数据入

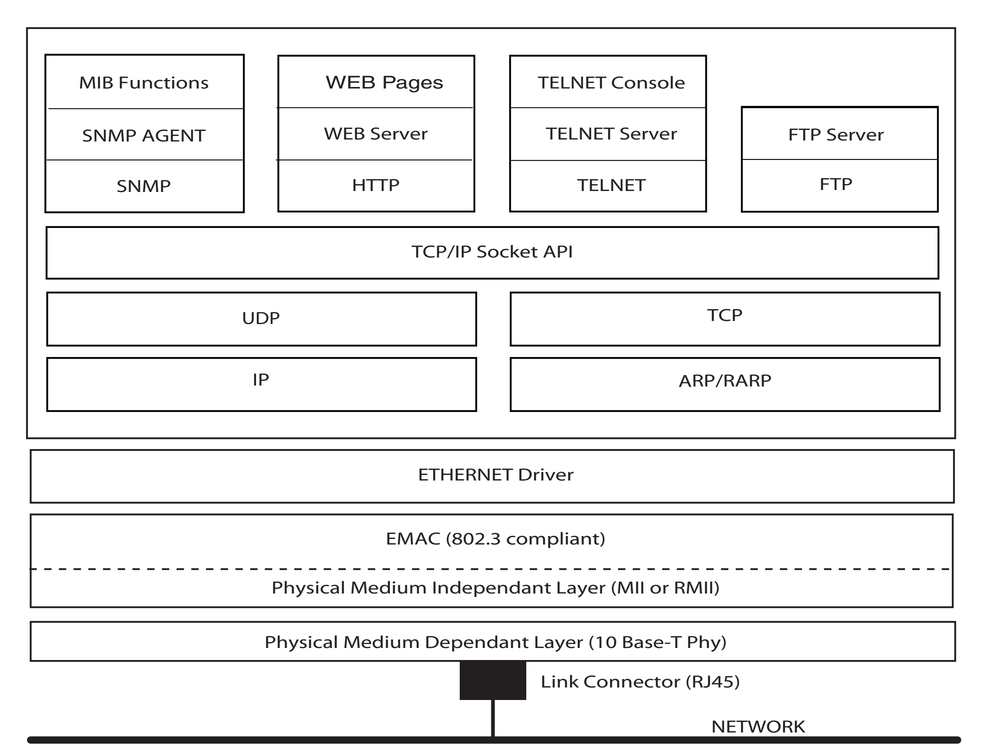
方框图

Figure 263. 方框图



应用框图

Figure 264. 以太网 MAC 应用框图



附属产品

I/O 线

用来连接外设的引脚可与 PIO 线复用。必须先对 PIO 控制器编程给 EMAC 引脚分配外设功能。RMII 模式下，未用引脚 (见 Table 101 : MII/RMII 信号映射图) 可作为普通 I/O 线使用。

电源管理

EMAC 可由电源管理控制器 (PMC) 定时，必须先配置 PMC 以使能 EMAC 时钟。

中断

EMAC有一条与高级中断控制器(AIC)连接的中断线 ,处理EMAC中断请求在配置EMAC前要对 AIC 编程。



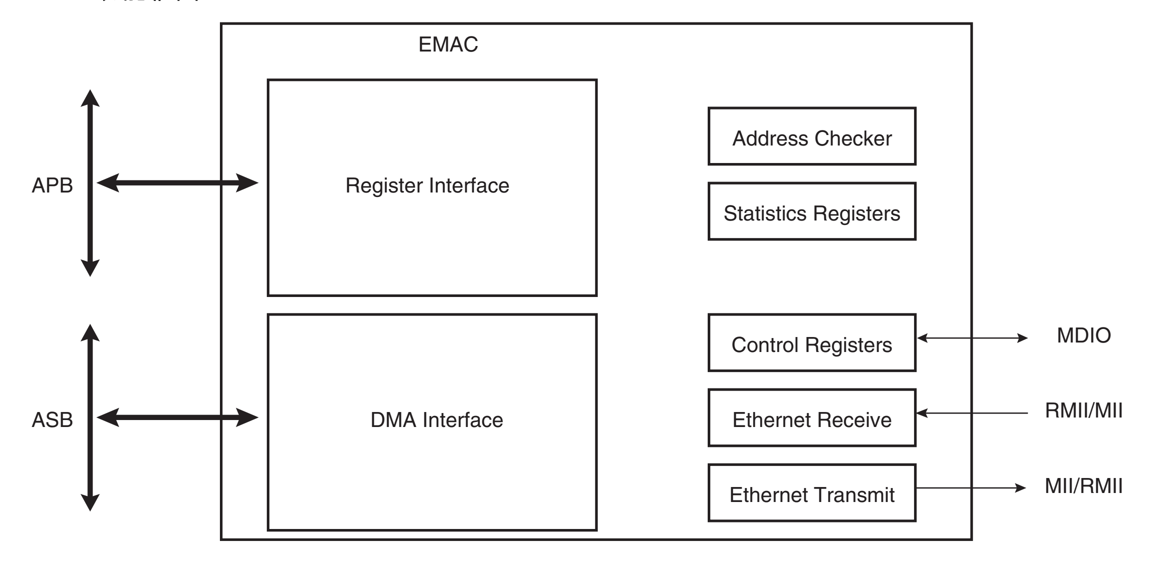
功能说明

以太网媒体访问控制 (EMAC) 引擎与 IEEE 802.3 以太网标准完全兼容。它管理帧发送与接收，包括冲突检测、报头产生与检测、 CRC 控制与产生及发送帧填充。

MAC 功能为 ：

- 帧封装与解封
- 错误检测
- 媒体访问管理 (MII、 RMII)

Figure 265. EMAC 功能框图



媒体独立接口

普通

以太网 MAC 可与 RMII 及 MII 接口连接。 ETH_CFG 寄存器中的 RMII 位控制选定的接口。当该位置位，选择 RMII 接口，否则选择 MII 接口。

MII与RMII接口均可实现IEEE 802.3u标准要求的10Mb/s与100Mb/s数据率。MII与RMII接口用到的信号见 Table 101。

Table 101. 引脚配置

引脚名称	MII	RMII
ETXCK_REFCK	ETXCK：发送时钟	REFCK：参考时钟
ECRS_ECRSDV	ECRS：载波敏感	ECRSDV：载波敏感 / 数据有效
ECOL	ECOL：冲突检测	
ERXDV	ERXDV：数据有效	
ERX0 - ERX3	ERX0 - ERX3：4 位接收数据	ERX0 - ERX1：2 位接收数据
ERXER	ERXER：接收错误	ERXER：接收错误
ERXCK	ERXCK：接收时钟	
ETXEN	ETXEN：发送使能	ETXEN：发送使能
ETX0-ETX3	ETX0 - ETX3：4 位发送数据	ETX0 - ETX1：2 位发送数据
ETXER	ETXER：发送错误	

RMII的目的是用缩减的引脚数来代替IEEE 802.3u MII。它使用2位进行发送(ETX0与ETX1)，2位进行接收 (ERX0 与 ERX1)。有一个发送使能 (ETXEN)，一个接收错误 (ERXER)，一个采博敏感 (ECRS_DV)，及一个对于 100Mb/s 数据率的 50 MHz 参考时钟 (ETXCK_REFCK)。

RMII 发送与接收操作

RMII与MII中使用相同信号。RMII使用引脚高效方式映射这些信号。发送与接收位由4位并行格式改为时钟速率加倍的 2 位并行格式。载波敏感与数据有效信号组合成 ECRS_ECRSDV 信号。. 该信号包含载波敏感、FIFO 状态及数据有效性。RMII 模式中未使用发送错误位 (ETXER) 及冲突检测位 (ECOL)。



发送 / 接收操作

标准 IEEE 802.3 包包括下列域 : 报头、帧启动分隔符 (SFD)、目的地址 (DA)、源地址 (SA)、长度、数据 (逻辑链接控制数据) 及帧校验序列 CRC32 (FCS)。

Table 102. 包格式

报头		帧 (1)					
交互 1s/0s	SFD	DA	SA	长度 / 类型	LLC Data	PAD	FCS
达到 7 字节	1 字节	6 字节	6 字节	2 字节			4 字节

Note: 帧长在 64 字节到 1518 字节间。

包为曼彻斯特编码与解码，并使用 NRZ 数据进行串行传输。所有域除数据域外长度固定。传输时， MAC 产生并挂起报头、 SFD 及 CRC 域

接收时将报头与 SFD 域解封。

报头与帧启动分隔符 (SFD)

报头用来从引入包得到同步位。当发送时，每个包包含 62 位交替的 1,0 报头。通过网络传输时，某些作为包传输的报头可能丢失。使用有两个连续 1 的帧启动分隔符 (SFD) 方式将字节对齐。

目的地址

目的地址 (DA) 指出包在网络上的目的，并滤除不希望得到的包。地址格式有三种类型 : 物理、端点传送及广播。物理地址是对应单节点的唯一地址，所有地址的 MSB 为 0。

多点传送地址开始时 MSB 为 1。 MAC 使用标准的 hashing 算法滤除所有映射为 6 位值的多点地址。该 6 位值编出一个 64 位阵列来滤除值。若地址值为全 1，则它为广播地址，表示包发送向所有节点。

源地址

源地址 (SA) 为发送包节点的物理地址。源地址不能成为多点发送或广播地址。该域被传输到缓冲存储器中。

长度 / 类型

若该域值小于或等于1500，则长度/类型域表示并发LLC数据域的字节数。若该域值大于1536，则，长度 / 类型域表示 MAC 客户协议种类 (协议类型)。

LLC 数据

数据域包含 46 到 1500 字节内容。若信息超过 1500 字节则需要分成多包。信息少于 46 字节则需要填充数据，将其加到 46 字节。若数据域填充，则长度域中表示有效数据字节。

FCS 域

帧检查序列 (FCS) 为一个 32 位 CRC 域，当接收包时允许对包计算与添加以检查错误。接收时，无错误包结果在 CRC 发生器中有指定格式。对于无正确 CRC 的包将拒收。

帧格式扩展

最初的以太网标准定义最小帧长为 64 字节 , 最大帧长为 1518 字节。这些数目包括从目的 MAC 地址域到帧检查序列域的所有字节。当开列帧长时未包括报头与启动帧分隔符域。 IEEE 802.3ac 标准将最大帧长扩展到 1522 字节 , 允许在以太网帧格式中插入 VLAN 标记符。 ETH_CFG 寄存器的 BIG 位帮助处理有 VLAN 标记符的包。

VLAN 协议允许在以太网帧格式中插入一个标识符 , 以辨别 VLAN 属于哪个帧。它允许帧从终端分配到逻辑组。它提供了多种好处 , 例如简化网络管理、允许形成工作组、提高网络安全性及提供一种限制广播域的方式 (参见 IEEE 标准 802.1Q 对 VLAN 协议的定义)。 802.3ac 标准仅定义了用于以太网的 VLAN 协议执行细节。

若目前 , 在源 MAC 地址域与长度域间插入 4 字节 VLAN 标记符 , 则前 2 字节的 VLAN 标记符包含 "802.1Q 标记符类型 " , 其值总为 0x8100。0x8100 值为保留的长度 / 类型域用来表示出现的 VLAN 标记符 , 而原来的长度 / 类型域在帧偏移四字节的位置。 VLAN 标记符的最后两字节包含下列信息 :

- 前三位为用户优先级域 , 可用来对以太网帧优先级进行分配。
- 下面一位为规范格式指示器 (CFI) , 用在以太网帧中表示含有路由信息域 (RIF)。
- 最后 12 位为 VLAN 标识符 (VID) 用来唯一标识 VLAN 属于哪一个以太网帧。

加入 VLAN 标识符后 , 802.3ac 标准将以太网的最大长度由 1518 字节扩展到 1522 字节。Table 103 给出 IEEE 802.3ac 标准下有 VLAN 标识符的以太网帧。

Table 103. 有 VLAN 标识符的以太网帧

报头	7 字节
帧起始分隔符	1 字节
目的 MAC 地址	6 字节
源 MAC 地址	6 字节
长度 / 类型 = 802.1Q 标识符类型	2 字节
标识符控制信息	2 字节
长度 / 类型	2 字节
MAC 客户数据	0 - n 字节
填充	0 - p 字节
帧检查序列	4 字节



DMA 操作

通过 DMA 接口在以太网 MAC 中传输数据。所有传输的均为 32 位字，可单、两次脉冲三或四字访问。脉冲访问不经过 16 字节边界。

DMA 控制器在 ASB 总线上执行四种类型操作。按照优先级排列，它们是：接收缓冲器管理器读、接收缓冲器管理器写、发送数据 DMA 读及发送数据 DMA 写。

发送器模式

发送帧数据要保存在相邻的存储位置中，它不需要字对齐。

用要发送的首字节地址对发送地址寄存器写入。

在发送控制寄存器中写入发送的字节数即可对发送初始化。

然后发送通道从存储器同时读取 32 位数据并将它们置于发送 FIFO。

当 FIFO 中载入三字后，发送块开始帧发送。

发送地址寄存器必须在发送控制寄存器前写入。当帧正在发送时，可以通过在发送地址与发送控制寄存器中写入新值来启动其它帧的发送。读发送地址寄存器将返回发送 FIFO 当前访问的缓冲器地址。

读发送控制寄存器将返回已发送字节总数。发送状态寄存器的 BNQ 位表示其它缓冲器可否正确排队。当该位置位时产生中断。

通过添加报头开始帧汇编并启动帧分隔符。数据从发送 FIFO 以字的形式取出。若需要可将正常增加到 60 字节。 CRC 以 32 位多项式计算。将它反转并添加的帧末尾处，使帧长最小为 64 字节。若发送控制寄存器的 NCRC 位置位，则 CRC 不能添加。

全双工模式下，立即发送帧。背靠背帧发送至少一次 96 位，以保证内部帧间隔。

半双工模式下，发送器检查载波敏感。若出现，则等到它失效，然后在 96 位时间内部帧间隔后开始发送。

若发送时出现冲突信号，发送器发送一个 32 位的堵塞序列到数据寄存器，并在补偿时间消逝后重新发送。若尝试 16 次均失败，则发出错误信号并放弃尝试。

若发送 DMA 欠载运行，坏的 CRC 使用与堵塞序列相同的机制最大添加。欠载运行还会引起 TXER 出现。

接收器模式

当收到包时，它将检查有效的报头、 CRC、校准、长度及地址。若符合所有标准，包成功存于接收缓冲器中。若接收到的 CRC 出错。则接收缓冲器恢复。每次收到含有 CRC 的帧将写入单接收缓冲器。

接收缓冲器是字对齐的并可包含 1518 或 1522 字节 (ETH_CFG 中 BIG = 1) 的数据 (以太网帧的最大长度)。

每个存于存储器的在接收缓冲器说明符列表中的接收帧的起始位置由接收好处队列指针寄存器指定。每个列表入口有两个字。第一个字为接收缓冲器地址，第二个字为接收状态。 Table 104 定义接收缓冲器说明符列表入口。

为接收帧，必须在每个列表入口的首个字位 [31:2] 写入正确的地址。字 0 的位 0 必须写入 0。

收到帧后，位 0 置位，第二个字表示引起帧复制到存储器的原因。接收缓冲器说明符列表应在接收使能前写入接收缓冲队列指针寄存器中 (通过设置网络控制寄存器的接收使能位)。一旦接收块开始将接收帧数据写入接收 FIFO，接收缓冲管理器读接收缓冲队列指针寄存器指向的第一个接收缓冲位置。若滤波器块激活，帧将复制到存储器中，接收数据 DMA 开始将数据写入接收缓冲器。若出现错误，缓冲器恢复。若收到的帧无错误，队列入口更新。缓冲器指针重新写入存储器，其低阶位用来表示帧成功接收及使用的缓冲器。下一个字为帧长及如何识别目的地址。下一个接收缓冲器位置则从下列字或，若当前缓冲器指针有环绕位设置，表的起始点处读取。在见到环绕位前缓冲器指针的最大数目为 1024。若未见到环绕位，假定环绕位在入口处。接收缓冲队列指针寄存器的低位必须置 0 以使能环绕功能正确工作。

若当接收缓冲管理器读接收缓冲器时位 0 置位，则缓冲器已用且在软件处理该帧并清除该位前不能使用。此时，DMA 块设置缓冲器接收状态寄存器不可用位并触发中断。将帧丢弃并若缓冲器可用时队列入口对重新接收的下一帧重读。每个丢弃的帧给统计寄存器值加一。当网络拥塞时，可对 MAC 编程提供反压。

这是当半双工模式冲突强制所有接收帧发送 64 位数据 (默认模式)。

读接收队列寄存器返回当前访问的队列入口位置。队列在每 1024 次入口后 (即 2048 个字) 或当回环位在入口第一个字的位 1 置位，又回到起点。

Table 104. 接收缓冲器说明符列表

位	功能
字 0	
31:2	接收缓冲器基地址
1	回环位。若该位置位，计数器与接收缓冲器队列指针寄存器相或来给出缓冲器使用后进入该表清零的指针。
0	所有权位。 1 表示软件拥有指针， 0 表示 DMA 拥有缓冲器。若当计数器读入口时该位不为零，接收状态寄存器缓冲器中的不可用位置位，并将计数器停止。
字 1	
31	全局为一广播地址检测
30	多点传送 hash 匹配
29	Unicast hash 匹配
28	外部地址 (可选)
27	未知源地址 (为将来使用保留)
26	本地地址匹配 (专用地址 1 匹配)
25	本地地址匹配 (专用地址 2 匹配)
24	本地地址匹配 (专用地址 3 匹配)
23	本地地址匹配 (专用地址 4 匹配)
22:11	保留，写入 0
10:0	包括 FCS 在内的帧长

地址校验

帧是否保存要根据网络配置寄存器中哪些使能、专用地址内容及 hash 寄存器以及帧目的地址。在该 MAC 执行中未校验帧源地址。

若在半双工模式下 MAC 发送时收到目的地址，帧将不复制到存储器。

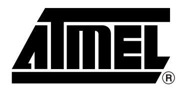
hash 寄存器长 64 位，在存储器中占据两个位置。

共有四个 48 位专用地址寄存器，每个占据两个存储器位置。第一个位置包括地址的前四字节；第二个位置包括地址的后两个字节。地址可保存为专用、组、本地、全局。

以太网帧以字节发送，低位在先。目的地址首位为 (及发送的第一位) 组 / 个体位，对于多点发送地址设置为 1，对于 unicast 设置为 0。该位对应专用地址寄存器首字的位 24。目的地址首字节的 MSB 对应专用地址寄存器的位 31。

专用地址寄存器与接收帧有效时的目的地址进行比较。复位或当首字节 [47:40] 写入时地址无效，激活或当最后字节 [7:0] 写入时地址有效。若接收帧地址与激活地址匹配，置位本地匹配信号并通过 HCLK 同步块将保存帧脉冲信号发送到 DMA 块。





若 unicast 或多点传送 hash 匹配出现，帧也将复制，广播地址为一，或网络配置寄存器的复制所有帧位置位。

广播地址为 0xFFFFFFFF 用来识别网络配置寄存器的无广播位是否为零。它设置广播匹配信号并触发保存帧信号。

网络配置寄存器的 unicast hash 使能及多点传送 hash 使能位使能 hash 匹配帧接收。因此，通过设置 hash 寄存器的所有位可接收所有的多点传送帧。

CRC 算法将目的地址由 6 位索引还原到 64 位 hash 寄存器中。若寄存器中等效位置位，帧是否匹配要由帧是多点传送还是 unicast 以及是否将适当的信号发送到 DMA 块上。若网络配置寄存器的复制所有帧位已置位，一旦收到目的地址，保存帧脉冲发送到 DMA 块上。

以太网 MAC (EMAC) 用户接口

Table 105. EMAC 寄存器映射

偏移	寄存器	寄存器名称	访问类型	复位值
0x00	EMAC 控制寄存器	ETH_CTL	读 / 写	0x0
0x04	EMAC 配置寄存器	ETH_CFG	读 / 写	0x800
0x08	EMAC 状态寄存器	ETH_SR	只读	0x6
0x0C	EMAC 发送地址寄存器	ETH_TAR	读 / 写	0x0
0x10	EMAC 发送控制寄存器	ETH_TCR	读 / 写	0x0
0x14	EMAC 发送状态寄存器	ETH_TSR	读 / 写	0x18
0x18	EMAC 接收缓冲队列指针	ETH_RBQP	读 / 写	0x0
0x1C	保留	–	只读	0x0
0x20	EMAC 接收状态寄存器	ETH_RSR	读 / 写	0x0
0x24	EMAC 中断状态寄存器	ETH_ISR	读 / 写	0x0
0x28	EMAC 中断使能寄存器	ETH_IER	只写	–
0x2C	EMAC 中断禁用寄存器	ETH_IDR	只写	–
0x30	EMAC 中断屏蔽寄存器	ETH_IMR	只读	0xFFF
0x34	EMAC PHY 维护寄存器	ETH_MAN	读 / 写	0x0
		统计寄存器 (1)		
0x40	帧发送成功寄存器	ETH_FRA	读 / 写	0x0
0x44	单冲突帧寄存器	ETH_SCOL	读 / 写	0x0
0x48	多冲突帧寄存器	ETH_MCOL	读 / 写	0x0
0x4C	帧接收成功寄存器	ETH_OK	读 / 写	0x0
0x50	帧检查下列错误寄存器	ETH_SEQE	读 / 写	0x0
0x54	对齐错误寄存器	ETH_ALE	读 / 写	0x0
0x58	延期发送帧寄存器	ETH_DTE	读 / 写	0x0
0x5C	最后冲突寄存器	ETH_LCOL	读 / 写	0x0
0x60	额外冲突寄存器	ETH_ECOL	读 / 写	0x0
0x64	载波敏感错误寄存器	ETH_CSE	读 / 写	0x0
0x68	发送欠载运行错误寄存器	ETH_TUE	读 / 写	0x0
0x6C	代码错误寄存器	ETH_CDE	读 / 写	0x0
0x70	长度超过错误寄存器	ETH_ELR	读 / 写	0x0
0x74	接收超时寄存器	ETH_RJB	读 / 写	0x0
0x78	过小帧寄存器	ETH_USF	读 / 写	0x0
0x7C	SQE 测试错误寄存器	ETH_SQEE	读 / 写	0x0
0x80	丢弃 RX 帧寄存器	ETH_DRFC	读 / 写	0x0
		地址寄存器		
0x90	EMAC Hash 地址高 [63:32]	ETH_HSH	读 / 写	0x0



Table 105. EMAC 寄存器映射

偏移	寄存器	寄存器名称	访问类型	复位值
0x94	EMAC Hash 地址低 [31:0]	ETH_HSL	读 / 写	0x0
0x98	EMAC 专用地址 1 低，前 4 字节	ETH_SA1L	读 / 写	0x0
0x9C	EMAC 专用地址 1 高，后 2 字节	ETH_SA1H	读 / 写	0x0
0xA0	EMAC 专用地址 2 低，前 4 字节	ETH_SA2L	读 / 写	0x0
0xA4	EMAC 专用地址 2 高，后 2 字节	ETH_SA2H	读 / 写	0x0
0xA8	EMAC 专用地址 3 低，前 4 字节	ETH_SA3L	读 / 写	0x0
0xAC	EMAC 专用地址 3 高，后 2 字节	ETH_SA3H	读 / 写	0x0
0xB0	EMAC 专用地址 4 低，前 4 字节	ETH_SA4L	读 / 写	0x0
0xB4	EMAC 专用地址 4 高，后 2 字节	ETH_SA4H	读 / 写	0x0

Note: 更多关于统计寄存器内容，见 Table 106 on page 584。

EMAC 控制寄存器

寄存器名称： ETH_CTL

访问类型： 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	BP

7	6	5	4	3	2	1	0
WES	ISR	CSR	MPE	TE	RE	LBL	LB

- **LB: 回送**

可选。设置时，回送信号为高电平。

- **LBL: 本地回送**

设置时，将ETX[3:0]连接到ERX[3:0]，ETXEN连接到ERXDV，强制全双工并驱动ERXCK与ETXCK_REFCK为MCK 4分频。

- **RE: 接收使能**

设置时，使能以太网 MAC 来接收数据。

- **TE: 发送使能**

设置时，使能以太网发送器来发送数据。

- **MPE: 管理端口使能**

设置为一以使能管理端口。当为零时强制 MDIO 为高阻态。

- **CSR: 清除统计寄存器**

该位只写。写一清除统计寄存器。

- **ISR: 增加统计寄存器**

该位只写。写一所有添加寄存器加一，为测试使用。

- **WES: 统计寄存器写使能**

设置为一以使统计寄存器可写，为测试使用。

- **BP: 反压**

若设置该位，则载半双工模式冲突时强制所有接收帧，通过发送 64 位数据 (默认模式)。



EMAC 配置寄存器

寄存器名称 : ETH_CFG

访问类型 : 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	RMII	RTY	CLK		EAE	BIG

7	6	5	4	3	2	1	0
UNI	MTI	NBC	CAF	–	BR	FD	SPD

- **SPD: 速度**

设置为 1 表示速度为 100 Mbit/s，设置为 0 则表示速度为 10 Mbit/s。没有其它效果。

- **FD: 全双工**

若设置为 1，发送块忽略冲突状态且载波敏感，并在发送时允许接收。

- **BR: 位速**

可选

- **CAF: 复制所有帧**

当设置为 1 时，接收所有有效帧。

- **NBC: 无广播**

当设置为 1 时，不接收全 1 的广播地址。

- **MTI: 多点传送 Hash 使能**

当设置时，当目的地址的 6 位指向的 hash 寄存器位置位，多点帧接收。

- **UNI: Unicast Hash 使能**

设置时，当目的地址的 6 位指向的 hash 寄存器位置位， unicast 帧接收。

- **BIG: 接收 1522 字节**

设置时， MAC 最多接收 1522 字节。通常 MAC 接收帧长为 1518 字节。

该位允许使用接收含 “VLAN tag” 的扩展以太网帧 (IEEE 802.3ac)。

- **EAE: 外部地址匹配使能**

可选。

- **CLK**

系统时钟 (MCK) 分频产生 MDC (MDIO 时钟)。为与 IEEE 标准 802.3 MDC 一致，不能超过 2.5 MHz。复位时，该域设置为 10，因此 MCK 被 32 分频。

CLK	MDC
00	MCK 8 分频
01	MCK 16 分频
10	MCK 32 分频
11	MCK 64 分频

- **RTY: 重新测试**

设置时，帧间为一个时隙。仅用于测试目的。普通操作时必须清零。

- **RMII: 缩减 MII**

设置时，该位使能 RMII 操作模式。复位时，选择 MII 模式。



EMAC 状态寄存器

寄存器名称 : ETH_SR

访问类型 : 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	–	–	–	IDLE	MDIO	LINK

- **LINK**

保留。

- **MDIO**

0 = MDIO 引脚未设置。

1 = MDIO 引脚设置。

- **IDLE**

0 = PHY 逻辑空闲。

1 = PHY 逻辑运行。

EMAC 发送地址寄存器

寄存器名称 : ETH_TAR

访问类型 : 读 / 写

31	30	29	28	27	26	25	24
ADDRESS							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
ADDRESS							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
ADDRESS							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ADDRESS							

- **ADDRESS: 发送地址寄存器**

写入发送帧地址，由发送 FIFO 读将要访问的缓冲器基地址。注意，若两个最低位不为零，在指定字节开始发送。

EMAC 发送控制寄存器

寄存器名称 : ETH_TCR

访问类型 : 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
NCRC	–	–	–	–	LEN		
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
LEN							

- **LEN: 发送帧长**

该寄存器写入将要发送的字节数目，不包括四个 CRC 字节，除非无 CRC 位出现。对这些位写入非零值，初始化发送。若值大于 1514 (若无 CRC，则为 1518)，发送一个超尺寸帧。该域可缓冲，以便在当前帧发送时，可有新帧排队。必须按照地址 - 长度的时序写入。读出值为要发送的字节总数 (即，帧发送时该值不会改变)。在两个 32 位字载入发送 FIFO 后，帧发送启动。长度必须足够大，以保证将两帧载入。

- **NCRC: 无 CRC**

若该位置位，即假设 CRC 包括在写入的长度中，且 MAC 不会在发送帧中添加 CRC。若缓冲器不够 64 字节，则发送一个短帧。该域可缓冲，以便在当前帧发送时，可有新帧排队。读出的是当前发送的帧值。

EMAC 发送状态寄存器

寄存器名称 : ETH_TSR

访问类型 : 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	UND	COMP	BNQ	IDLE	RLE	COL	OVR

- **OVR: 以太网发送缓冲溢出**

当 BNQ 未置位时，软件写入发送地址寄存器 (ETH_TAR) 或发送控制寄存器 (ETH_TCR)。该位写 1 清除。

- **COL: 冲突出现**

设置冲突出现。该位写 1 清除。

- **RLE: 超出限制重试**

该位写 1 清除。

- **IDLE: 发送器空闲**

发送器无帧发送时出现。当发送控制寄存器中的一部分发送帧写入时清除。该位只读。

- **BNQ: 以太网发送缓冲器无队列**

设置时，软件可写入新缓冲器地址及发送 DMA 控制器长度。有帧发送就绪或有其它正在发送帧时清除。该位只读。

- **COMP: 发送结束**

当帧发送完成后设置。该位写 1 清除。

- **UND: 发送欠载运行**

当发送 DMA 不能及时从存储器中读取数据时设置。此时，发送器强制破坏 CRC。该位写 1 清除。



EMAC 接收缓冲队列指针寄存器

寄存器名称： ETH_RBQP

访问类型： 读 / 写

31	30	29	28	27	26	25	24
ADDRESS							

23	22	21	20	19	18	17	16
ADDRESS							

15	14	13	12	11	10	9	8
ADDRESS							

7	6	5	4	3	2	1	0
ADDRESS							

- **ADDRESS: 接收缓冲队列指针**

写入接收队列起始地址。读出当前使用缓冲器指针。计数器缓冲器强制字对齐。

EMAC 接收状态寄存器

寄存器名称： ETH_RSR

访问类型： 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	OVR	REC	BNA

- **BNA: 缓冲器无效**

尝试得到新缓冲器，指针表示它属于处理器。每次新帧开始时 DMA 重读指针，直到找到有效指针。每次尝试失败后该位置位。该位写 1 清除。

- **REC: 帧接收**

已收到一个或多个帧，并置于存储器中。该位写 1 清除。

- **OVR: RX 溢出**

DMA 块不能将收到帧保存到存储器中，不管是由于 ASB 总线确认不及时，还是因为没有返回 OK HRESP。此时，缓冲器恢复。该位写 1 清除。

EMAC 中断状态寄存器

寄存器名称 : ETH_ISR

访问类型 : 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	ABT	ROVR	LINK	TIDLE

7	6	5	4	3	2	1	0
TCOM	TBRE	RTRY	TUND	TOVR	RBNA	RCOM	DONE

- **DONE: 管理完成**

PHY 维护寄存器完成其操作。读时清除。

- **RCOM: 接收完成**

帧已存入存储器。读时清除。

- **RBNA: 接收缓冲器无效**

读时清除。

- **TOVR: 发送缓冲器溢出**

当发送状态寄存器 (ETH_TSR) 的 BNQ 位未置位，软件写发送地址寄存器 (ETH_TAR) 或发送控制寄存器 (ETH_TCR)。读时清除。

- **TUND: 发送缓冲器欠载运行**

以太网发送缓冲器欠载运行。发送 DMA 未能及时取得发送的帧数据。读时清除。

- **RTRY: 重试限制**

重试限制超出。读时清除。

- **TBRE: 发送缓冲寄存器空**

软件在发送 DMA 控制器中写入新缓冲器地址及长度。有帧发送就绪并有其它帧正在发送时清除。读时清除。

- **TCOM: 发送完成**

当帧发送完成后置位。读时清除。

- **TIDLE: 发送空闲**

当所有帧发送完成后置位。读时清除。

- **LINK**

当 LINK 引脚改变值时置位。可选。

- **ROVR: RX 溢出**

当 RX 溢出状态位置位时置位。读时清除。

- **ABT: 中止**

当 DMA 传输时中止出现时置位。读时清除。



EMAC 中断使能寄存器

寄存器名称： ETH_IER

访问类型： 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	ABT	ROVR	LINK	TIDLE
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TCOM	TBRE	RTRY	TUND	TOVR	RBNA	RCOM	DONE

- **DONE: 管理完成中断使能**
- **RCOM: 接收完成中断使能中断使能**
- **RBNA: 接收缓冲器无效中断使能**
- **TOVR: 发送缓冲器溢出中断使能**
- **TUND: 发送缓冲器欠载运行中断使能**
- **RTRY: 重试限制中断使能**
- **TBRE: 发送缓冲寄存器空中断使能**
- **TCOM: 发送完成中断使能**
- **TIDLE: 发送空闲中断使能**
- **LINK: LINK 中断使能**
- **ROVR: RX 溢出中断使能**
- **ABT: 中止中断使能**

0：无效。

1：使能相应中断。

EMAC 中断禁用寄存器

寄存器名称 : ETH_IDR

访问类型 : 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	ABT	ROVR	LINK	TIDLE

7	6	5	4	3	2	1	0
TCOM	TBRE	RTRY	TUND	TOVR	RBNA	RCOM	DONE

- **DONE: 管理完成中断禁用**
- **RCOM: 接收完成中断使能中断禁用**
- **RBNA: 接收缓冲器无效中断禁用**
- **TOVR: 发送缓冲器溢出中断禁用**
- **TUND: 发送缓冲器欠载运行中断禁用**
- **RTRY: 重试限制中断禁用**
- **TBRE: 发送缓冲寄存器空中断禁用**
- **TCOM: 发送完成中断禁用**
- **TIDLE: 发送空闲中断禁用**
- **LINK: LINK 中断禁用**
- **ROVR: RX 溢出中断禁用**
- **ABT: 中止中断禁用**

0 : 无效。

1 : 禁用相应中断。

EMAC 中断屏蔽寄存器

寄存器名称： ETH_IMR

访问类型： 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	ABT	ROVR	LINK	TIDLE
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TCOM	TBRE	RTRY	TUND	TOVR	RBNA	RCOM	DONE

- **DONE: 管理完成中断屏蔽**
- **RCOM: 接收完成中断使能中断屏蔽**
- **RBNA: 接收缓冲器无效中断屏蔽**
- **TOVR: 发送缓冲器溢出中断屏蔽**
- **TUND: 发送缓冲器欠载运行中断屏蔽**
- **RTRY: 重试限制中断屏蔽**
- **TBRE: 发送缓冲寄存器空中断屏蔽**
- **TCOM: 发送完成中断屏蔽**
- **TIDLE: 发送空闲中断屏蔽**
- **LINK: LINK 中断屏蔽**
- **ROVR: RX 溢出中断屏蔽**
- **ABT: 中止中断屏蔽**

0：相应中断禁用。

1：相应中断使能。

重要提示： 当相应位置位时中断禁用。由于通常屏蔽位设置使能中断，对于 AT91 是不标准的。

EMAC PHY 维护寄存器

寄存器名称： ETH_MAN

访问类型： 读 / 写

31	30	29	28	27	26	25	24
LOW	HIGH	RW		PHYA			

23	22	21	20	19	18	17	16
PHYA	REGA					CODE	

15	14	13	12	11	10	9	8
DATA							

7	6	5	4	3	2	1	0
DATA							

对该寄存器写入启动移位寄存器，控制与 PHY 的串行连接。在每个移位周期， MDIO 引脚等于移位寄存器的 MSB，而移位寄存器的 LSB 等于 MDIO 引脚值。当移位完成产生中断，并设置网络状态寄存器的 IDLE 域。

当读取时，给出当前移出值。

- **DATA**

对于写操作，是要写入 PHY 的数据。读操作后，它的数据是由 PHY 读取的数据。

- **CODE**

根据 IEEE 标准 802.3 必须写入 10。读与写相同。

- **REGA**

寄存器地址。指定 PHY 访问寄存器。

- **PHYA**

PHY 地址。通常为 0。

- **RW**

读 / 写操作。 10 为读； 01 为写。其它值对于 PHY 管理帧无效。

- **HIGH**

必须写入 1，以保证 PHY 管理帧有效。与 IEEE 标准 802.3 一致。

- **LOW**

必须写入 0，以保证 PHY 管理帧有效。与 IEEE 标准 802.3 一致。

EMAC Hash 地址高寄存器

寄存器名称： ETH_HSH

访问类型： 读 / 写

31	30	29	28	27	26	25	24
ADDR							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
ADDR							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
ADDR							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ADDR							

- **ADDR**

Hash 地址位 63 到 32。

EMAC Hash 地址低寄存器

寄存器名称： ETH_HSL

访问类型： 读 / 写

31	30	29	28	27	26	25	24
ADDR							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
ADDR							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
ADDR							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ADDR							

- **ADDR**

Hash 地址位 31 到 0。

EMAC 专用地址 (1, 2, 3, 4) 高寄存器

寄存器名称 : ETH_SA1H,...ETH_SA4H

访问类型 : 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
ADDR							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ADDR							

- **ADDR**

Unicast 地址 (1, 2, 3 , 4), 位 47:32。

EMAC 专用地址 (1, 2, 3, 4) 低寄存器

寄存器名称 : ETH_SA1L,...ETH_SA4L

访问类型 : 读 / 写

31	30	29	28	27	26	25	24
ADDR							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
ADDR							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
ADDR							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ADDR							

- **ADDR**

Unicast 地址 (1, 2, 3 , 4), 位 31:0。



EMAC 统计寄存器块寄存器

这些寄存器在读时复位为 0，当计数到最大值时，所有值为 1。应经常对其读取，以防止数据丢失。

统计寄存器块包括的寄存器见 Table 106。

Table 106. 统计寄存器块

寄存器	寄存器名称	说明
帧发送完成寄存器	ETH_FRA	一个 24 位寄存器，用来计算成功发送的帧数目。
单冲突帧寄存器	ETH_SCOL	一个 16 位寄存器，用来计算在发送前经历单冲突及经历载波丢失或欠载运行的帧数目。
多冲突帧寄存器	ETH_MCOL	一个 16 位寄存器，用来计算发送前经历 2 到 15 次冲突的帧数目 (62 - 1518 字节，无载波丢失，无欠载运行)。
帧接收完成寄存器	ETH_OK	一个 24 位寄存器，用来计算好的接收帧数目，即地址认可。好帧长度为 64 到 1518 字节，且无 FCS、对齐或代码错误。
帧校验序列错误寄存器	ETH_SEQE	一个 8 位寄存器，用来计算帧地址可辨、整数字节、有错 CRC 还是 64 到 1518 字节长。
对齐错误寄存器	ETH_ALE	一个 8 位寄存器，用来计算帧： - 帧地址可辨、 - 非整数字节、 - 当其长度缩减到整数时有错误的 CRC、 - 64 到 1518 字节长。
延迟发送帧寄存器	ETH_DTE	一个 16 位寄存器，用来计算由于首次发送时载波敏感而造成帧延迟的数目 (无欠载运行或冲突)。
最后冲突寄存器	ETH_LCOL	一个 8 位寄存器，用来计算在时隙 (512 位) 出现后经历冲突的帧数。无载波丢失或欠载运行。最后冲突将计算两次，即冲突与最后冲突各一次。
额外冲突寄存器	ETH_ECOL	一个 8 位寄存器，用来计算由于出现 16 次冲突而造成的帧发送失败 (64 - 1518 字节，无欠载运行或冲突)。
载波敏感错误寄存器	ETH_CSE	一个 8 位寄存器，用来计算未检测到载波敏感或开始发送后仍保持一个时隙 (512 位) 半双工模式的帧数 (无额外冲突)。
发送欠载运行寄存器	ETH_TUE	一个 8 位寄存器，用来计算由于发送 DMA 欠载运行而未能发送的帧数。若该寄存器增加，则其它寄存器值不会增加。
代码错误寄存器	ETH_CDE	一个 8 位寄存器，用来计算地址可辨，接收时有 RXER 出现的帧数。若该寄存器增加，则其它寄存器值不会增加。
长度超过错误寄存器	ETH_ELR	一个 8 位寄存器，用来计算收到的长度超过 1518 字节，但即无 CRC 错误，也无对齐或代码错误的帧数。
接收超时寄存器	ETH_RJB	一个 8 位寄存器，用来计算收到的长度超过 1518 字节，但有 CRC 错误，或有对齐、代码错误的帧数。
过小帧寄存器	ETH_USF	一个 8 位寄存器，用来计算收到的长度小于 64 字节，但即无 CRC 错误，也无对齐或代码错误的帧数。
SQE 测试错误寄存器	ETH_SQEE	一个 8 位寄存器，用来计算在引脚 ETXEN 失效的一个时隙内引脚 ECOL 未有效的帧数。
丢弃 RX 帧寄存器	ETH_DRFC	一个 16 位寄存器，用来计算地址可辨，但由于接收缓冲器有效而使帧不能复制到存储器的帧数。

AT91RM9200 电气特性

绝对极限值

Table 107. 绝对极限值 *

工作温度 (工业级)	-40° C 到 +85° C
存储温度	-60°C 到 +150°C
各个输入引脚对地电压	-0.3V 到 +3.6V
最大工作电压 (V_{DDCORE}、 V_{DDPLL} 及 V_{DDOSC})	1.95V
最大工作电压 (V_{DDIOM} 及 V_{DDIOP})	3.6V
直流输出电流 (SDA10、 SDCKE、 SDWE、 RAS、 CAS)	16 mA
直流输出电流 (其它引脚)	8 mA

*NOTICE: 如果强制芯片在超出“ 绝对极限值” 表中所列的条件之下工作可能造成器件的永久损坏。这仅是工作应力的极限。并不表示器件可以工作于表中所列条件之下，或是那些超越工作范围明确规定的其他条件之下。长时间工作于绝对极限值可能会影响器件的寿命。



直流特性

下列特性适用的温度范围：T_A = -40°C 到 85°C，除非另有说明，结温达到 T_J = 100°C。

Table 108. 直流特性

符号	参数	条件		最小值	典型值	最大值	单位
V_{DDCORE}	内核直流电压			1.65		1.95	V
V_{DDOSC}	振荡器直流电压			1.65		1.95	V
V_{DDPLL}	PLL 直流电压			1.65		1.95	V
V_{DDIOM}	存储器 I/O 直流电压			V_{DDCORE}		V_{DDCORE} + 1.5 或 3.6	V
V_{DDIOP}	外设 I/O 直流电压			V_{DDCORE}		V_{DDCORE} + 1.5 或 3.6	V
V_{IL}	输入低电平电压			-0.3		0.8	V
V_{IH}	输入高电平电压			2		V_{DD} + 0.3(1)	V
V_{OL}	输出低电平电压	SDA10, SDCKE, SDWE, RAS, CAS 引脚： I_{OL} = 16 mA(2) I_{OL} = 0 mA(2)				0.4 0.2	V
		其它引脚： I_{OL} = 8 mA(2) I_{OL} = 0 mA(2)				0.4 0.2	
V_{OH}	输出高电平电压	SDA10, SDCKE, SDWE, RAS, CAS 引脚： I_{OH} = 16 mA(2) I_{OH} = 0 mA(2)		V_{DD} - 0.4(1) V_{DD} - 0.2(1)			V
		其它引脚： I_{OH} = 8 mA(2) I_{OH} = 0 mA(2		V_{DD} - 0.4(1) V_{DD} - 0.2(1)			
I_{LEAK}	输入漏电流	上拉电阻禁用				1	µA
I_{PULL}	输入上拉电流	V_{DD} = 3.0V(1), V_{IN} = 0		129			µA
		V_{DD} = 3.6V(1), V_{IN} = 0				322	
C_{IN}	输入电容	208-PQFP 封装				8.8	pF
		256-LFBGA 封装				7.6	
I_{SC}	静电流	V_{DDCORE} = 2V, MCK = 0 Hz	T_A = 25°C		179	1157	µA
		所有输入驱动 TMS, TDI, TCK, NRST = 1	T_A = 85°C		1610	7989	

Notes: 1. V_{DD} 可用于 V_{DDIOM}, V_{DDIOP}, V_{DDPLL} 及 V_{DDOSC}。
2. I_O = 输出电流。

时钟特性

参数在下列条件下给定：

- V_{DDCORE} = 1.8V
- 环境温度 = 25°C

温度降额因子见"Temperature Derating Factor" on page 604说明，而V_{DDCORE} 电压降额因子见"VDDCORE Voltage Derating Factor" on page 604 说明。

处理器时钟特性

Table 109. 处理器时钟波形参数

符号	参数	条件	最小值	最大值	单位
$1/(t_{CPPCK})$	处理器时钟频率			209.0	MHz
t_{CPPCK}	处理器时钟周期		4.8		ns
t_{CHMCK}	主机时钟高半周期		2.2		ns
t_{CLMCK}	主机时钟低半周期		2.2		ns

主机时钟特性

Table 110. 主机时钟波形参数

符号	参数	条件	最小值	最大值	单位
$1/(t_{CPMCK})$	主机时钟频率			80.0	MHz
t_{CPMCK}	主机时钟周期		12.5		ns
t_{CHMCK}	主机时钟高半周期		6.3		ns
t_{CLMCK}	主机时钟低半周期		6.3		ns

XIN 时钟特性 (1)

Table 111. XIN 时钟电气特性

符号	参数	条件	最小值	最大值	单位
$1/(t_{CPXIN})$	XIN 时钟频率			50.0	MHz
t_{CPXIN}	XIN 时钟周期		20.0		ns
t_{CHXIN}	XIN 时钟高半周期		0.4 x t_{CPXIN}	0.6 x t_{CPXIN}	
t_{CLXIN}	XIN 时钟低半周期		0.4 x t_{CPXIN}	0.6 x t_{CPXIN}	
C_{IN}	XIN 输入电容	Note (1)		25	pF
R_{IN}	XIN 下拉电阻	Note (1)		500	kOhm

Notes: 1. 这些特性仅适用于主振荡器为旁路模式时(即CKGR_MOR寄存器的MOSCEN = 0，见See "PMC Clock Generator Main Oscillator Register" on page 276.)。



功耗

Table 112 与 Table 113 值估计如下工作条件下功耗：

- V_{DDIO} = 3.3V
- V_{DDCORE} = V_{DDPLL} = V_{DDOSC} = 1.8V
- T_A = 25° C
- MCK = 60 MHz
- PCK = 180 Mhz
- SLCK = 32.768 kHz

这些图表示在 V_{DDCORE} 上测得的功耗。

Table 112. PMC 模式功耗 (1)

模式	状态	消耗	单位
普通	ARM 内核时钟使能。 所有外设时钟无效。	31.7	mA
空闲	ARM 内核时钟禁用并等待下一个中断。 所有外设时钟无效。	15.0	
慢时钟	主振荡器与 PLL 关闭。 处理器与所有外设运行在慢时钟下。	1.7	
Standby	空闲与慢时钟模式的组合。	1.7	

Note: 1. 内部 SRAM 中代码。

Table 113. 外设功耗 (1)

外设	功耗	单位
PIO 控制器	0.6	mA
USART	1.6	
MCI	1.9	
UDP	1.5	
TWI	0.4	
SPI	1.4	
SSC	1.8	
定时器计数器通道	0.4	
UHP	3.4	
EMAC	4.3	
PMC PLL(2) 慢时钟振荡器 (3) 主振荡器 (3)	 3144 858 350	 uA nA uA

Notes: 1. 内部 SRAM 中代码。
2. V_{DDPLL} 电源下功耗。
3. V_{DDOSC} 电源下功耗。

晶振特性

32 kHz 振荡器特性

Table 114. 32 kHz 振荡器特性

符号	参数	状态	最小值	典型值	最大值	单位
$1/(t_{CP32KHz})$	晶振频率			32.768		kHz
	占空比	在 PCK 输出引脚测得	40	50	60	%
t_{ST}	启动时间	V_{DDOSC} = 1.8V R_s = 50 kΩ, C_L = 12.5 pF[1]			900	ms

Note: 1. R_s 为等效串联电阻， C_L 为等效负载电容。

主振荡器特性

Table 115. 主振荡器特性

符号	参数	状态	最小值	典型值	最大值	单位
$1/(t_{CPMAIN})$	晶振频率		3	16	20	MHz
C_{L1}, C_{L2}	内部负载电容 (C_{L1} = C_{L2})			25		pF
C_L	等效负载电容	C_{L1} = C_{L2} = 25 pF		12.5		pF
	占空比	在 PCK 输出引脚测得	40	50	60	%
t_{ST}	启动时间	V_{DDPLL} = 1.8V $1/(t_{CPMAIN})$ = 3 MHz 主振荡器引脚未连接电容 (XIN 与 XOUT)			14.5	ms

PLL 特性

Table 116. 锁相环特性

符号	参数	状态	最小值	典型值	最大值	单位
F_{OUT}	输出频率		80		240	MHz
F_{IN}	输入频率		1		32	MHz
K_O	VCO 增益		120	190	300	MHz/V
I_P	电流消耗		36	44	60	µA

计数器特性

电气特性

Table 117. 电参数

符号	参数	条件	最小值	典型值	最大值	单位
输入电平						
V_{IL}	低电平				0.8	V
V_{IH}	高电平		2.0			V
V_{DI}	差分输入	|(D+) - (D-)|	0.2			V
V_{CM}	差分输入共模范围		0.8		2.5	V
C_{IN}	发送器电容	每条线到地电容			20	pF
I	Hi-Z 状态数据线漏	0V < V_{IN} < 3.3V	-5		+5	µA
R_{EXT}	推荐的外部 USB 串行电阻	串行时在每个 USB 引脚上为 ±5%		27		
输出电平						
V_{OL}	低电平输出	由 RL 在 1.425 kOhm 3.6V 的条件下测量			0.3	V
V_{OH}	高电平输出	由 RL 在 1.425 kOhm GND 的条件下测量	2.8			V
V_{CRS}	输出信号交叉电压	测量条件见 Figure 266	1.3		2.0	V

切换特性

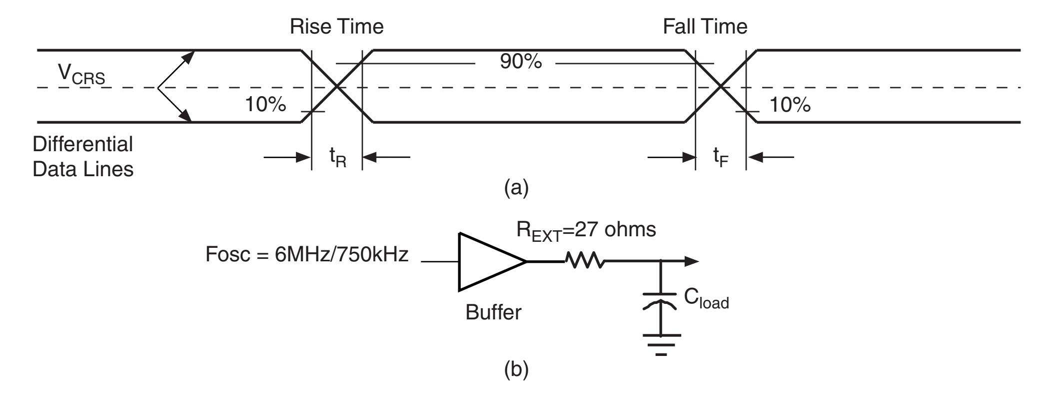
Table 118. 慢模式下

符号	参数	条件	最小值	典型值	最大值	单位
t_{FR}	上升时间	C_{LOAD} = 400 pF	75		300	ns
t_{FE}	下降时间	C_{LOAD} = 400 pF	75		300	ns
t_{FRFM}	上升 / 下降时间匹配	C_{LOAD} = 400 pF	80		120	%

Table 119. 全速模式下

符号	参数	条件	最小值	典型值	最大值	单位
t_{FR}	上升时间	C_{LOAD} = 50 pF	4		20	ns
t_{FE}	下降时间	C_{LOAD} = 50 pF	4		20	ns
t_{FRFM}	上升 / 下降时间匹配		90		111.11	%

Figure 266. USB 数据信号上升与下降时间



AT91RM9200 交流特性

适用条件及降额数据

条件与定时计算

所有给出的延迟是下列条件下的典型值：

- V_{DDIOM} = 3.3V
- V_{DDCORE} = 1.8V
- 环境温度 = 25°C
- 负载电容 = 0 pF
- 输出电平变化检测为 (0.5 x V_{DDIOM})
- 低电平检测时输入电平为 (0.3 x V_{DDIOM}) ；高电平检测输入电平为 (0.7 x V_{DDIOM})

交流特性表中给出的最小最大值考虑了加工变化及设计。为获得其它条件下定时，使用下列等式：

$$t = \delta_{T^\circ} \times \left((\delta_{VDDCORE} \times t_{DATASHEET}) + \left(\delta_{VDDIOM} \times \sum C_{SIGNAL} \times \delta_{CSIGNAL} \right) \right)$$

where:

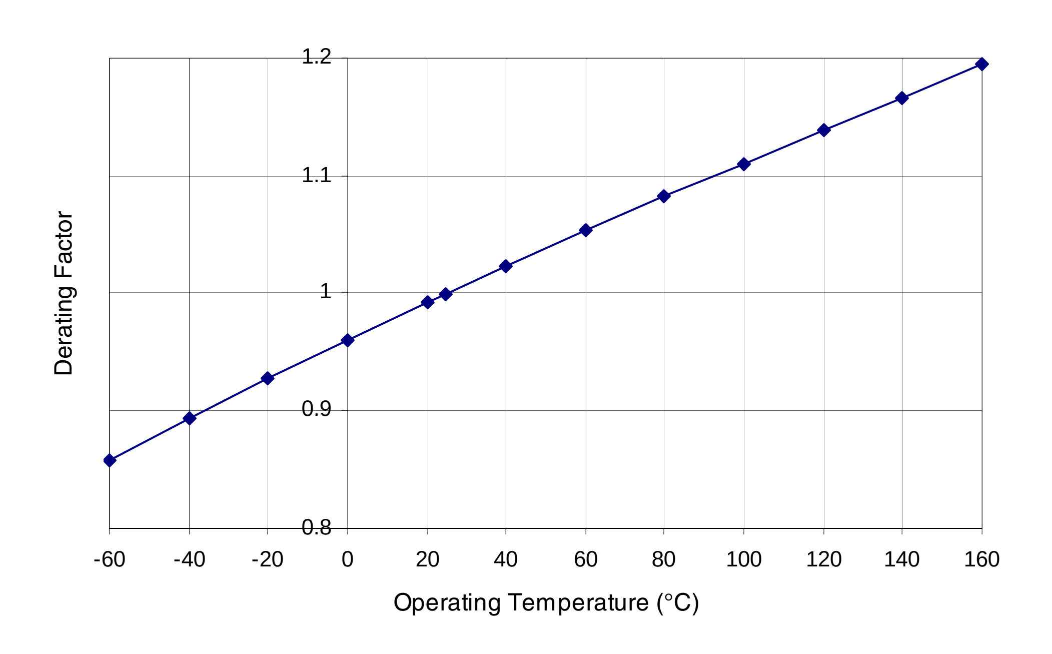
- δ_{T° 为温度降额因子，见 Figure 267 on page 594。
- $\delta_{VDDCORE}$ 为内核电源降额因子，见 Figure 268 on page 594。
- $t_{DATASHEET}$ 为负载电容为 0 pF 时的最小或最大定时值。
- δ_{VDDIOM} 为 IO 电源降额因子，见 Figure 269 on page 595。
- C_{SIGNAL} 为输出引脚容性负载 (1)。
- $\delta_{CSIGNAL}$ 为根据手册中相关输出引脚容性负载最小或最大值的降额因子。

输入延迟为典型值。

Note: 1. 用户必须考虑封装负载贡献 (C_{IN})，见 Table 109, “DC Characteristics,” on page 596。

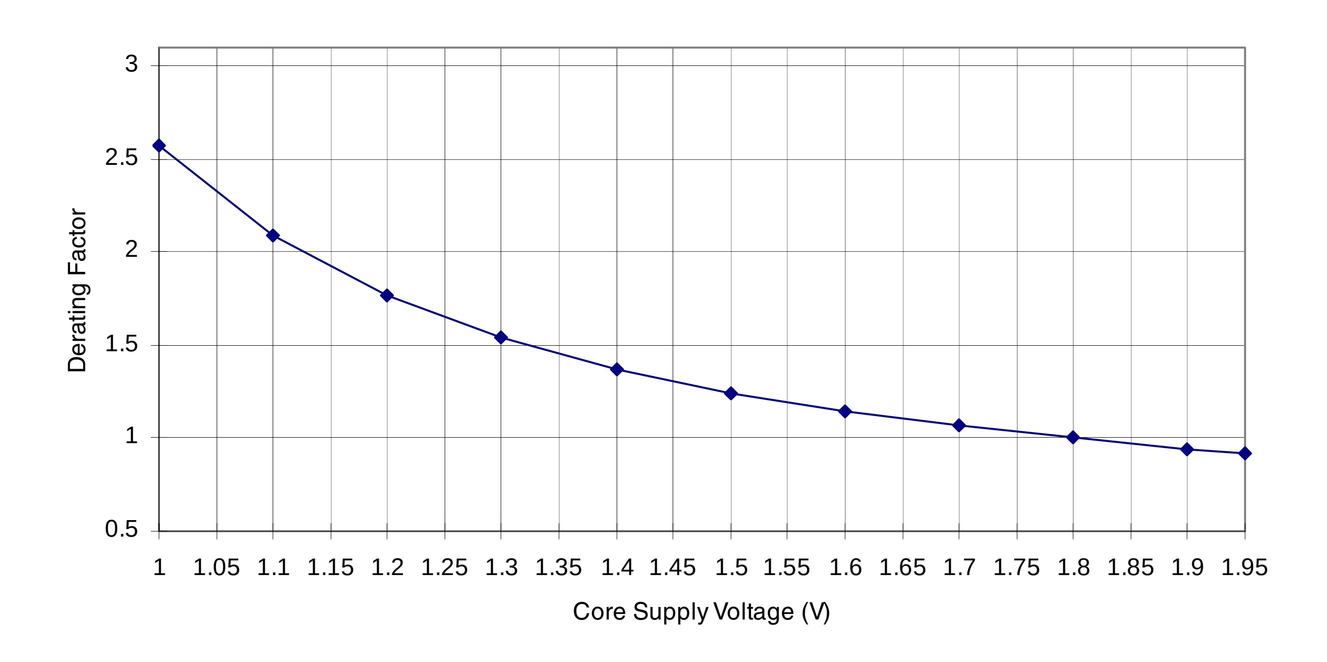
温度降额因子

Figure 267. 不同工作温度下降额曲线



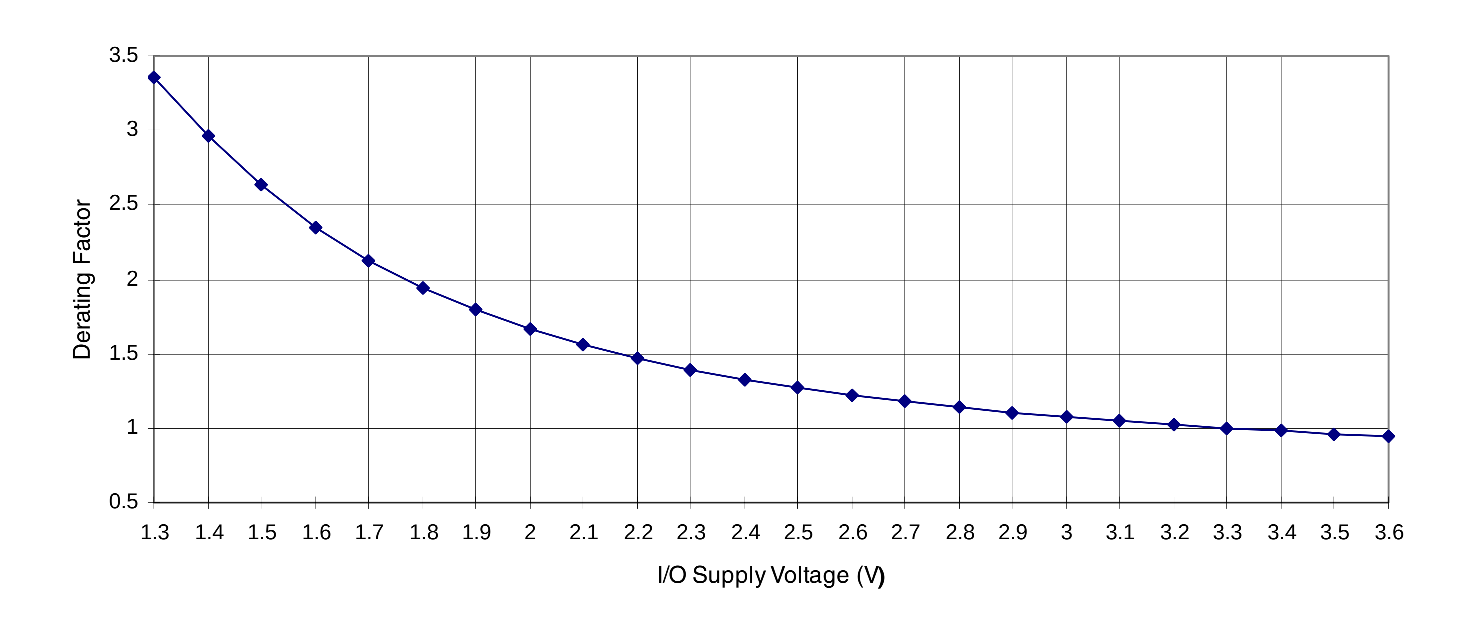
V_{DDCORE} 电压降额因子

Figure 268. 不同内核电压下降额因子



V_{DDIOM} 电压降额因子

Figure 269. 不同 IO 电压下降额曲线



Note: 本例中降额因子仅适用于对相关输出引脚定时。

EBI 定时

与 MCK 相关的 SMC 信号

Table 120、 Table 121 及 Table 122 给出工作在 “Ã ºž”Î¹®þ±ºýÀ„” on page 593 中定义的环境中的相关定时。

Table 120. 通用功能 SMC 信号

符号	参数	条件	最小值	最大值	单位
SMC_1	MCK 下降到 NUB 有效	C_{NUB} = 0 pF	5.0	7.5	ns
		C_{NUB} 降额	0.028	0.045	ns/pF
SMC_2	MCK 下降到 NLB/A0 有效	C_{NLB} = 0 pF	4.9	7.5	ns
		C_{NLB} 降额	0.028	0.045	ns/pF
SMC_3	MCK 下降到 A1 - A25 有效	C_{ADD} = 0 pF	4.9	7.4	ns
		C_{ADD} 降额	0.028	0.045	ns/pF
SMC_4	MCK 下降到片选改变 (无地址到片选设置)	C_{NCS} = 0 pF	4.3	6.5	ns
		C_{NCS} 降额	0.028	0.045	ns/pF
SMC_5	MCK 下降到片选激活 (地址到片选设置) (1)	C_{NCS} = 0 pF	$(nacss \times t_{CPMCK}) + 4.3$ (2)	$(nacss \times t_{CPMCK}) + 6.5$ (2)	ns
		C_{NCS} 降额	0.028	0.045	ns/pF
SMC_6	片选无效到 MCK 下降 (地址到片选设置) (1)	C_{NCS} = 0 pF	$(nacss \times t_{CPMCK}) + 4.4$ (2)	$(nacss \times t_{CPMCK}) + 6.5$ (2)	ns
		C_{NCS} 降额	0.028	0.045	ns/pF
SMC_7	NCS 最小脉宽 (地址到片选设置) (1)	C_{NCS} = 0 pF	$(((n + 2) - (2 \times nacss)) \times t_{CPMCK})$ (2) (3)		ns
SMC_8	NWAIT 最小脉宽 (1)		t_{CPMCK}		ns

Notes: 1. 降额因子不适用于 t_{CPMCK}。
2. nacss = 插入的地址到片选设置周期数。
3. n = 插入的标准等待状态数。

Table 121. SMC 写信号

符号	参数	条件	最小值	最大值	单位
SMC_{10}	MCK 上升到 NWR 激活 (无等待状态) (5)	C_{NWR} = 0 pF	4.8	7.2	ns
		C_{NWR} 降额	0.028	0.045	ns/pF
SMC_{11}	MCK 上升到 NWR 激活 (等待状态)	C_{NWR} = 0 pF	4.8	7.2	ns
		C_{NWR} 降额	0.028	0.045	ns/pF
SMC_{12}	MCK 下降到 NWR 无效 (无等待状态) (5)	C_{NWR} = 0 pF	4.8	7.2	ns
		C_{NWR} 降额	0.028	0.045	ns/pF
SMC_{13}	MCK 上升到 NWR 无效 (等待状态)	C_{NWR} = 0 pF	4.8	7.2	ns
		C_{NWR} 降额	0.028	0.045	ns/pF
SMC_{14}	MCK 上升到 D0 - D15 输出有效	C_{DATA} = 0 pF	4.1	7.9	ns
		C_{DATA} 降额	0.028	0.044	ns/pF
SMC_{15}	NWR 高到 NUB 改变 (5)	C_{NUB} = 0 pF	3.4		ns
		C_{NUB} 降额	0.028		ns/pF
SMC_{16}	NWR 高到 NLB/A0 改变 (5)	C_{NLB} = 0 pF	3.7		ns
		C_{NLB} 降额	0.028		ns/pF
SMC_{17}	NWR 高到 A1 - A25 改变 (5)	C_{ADD} = 0 pF	3.3		ns
		C_{ADD} 降额	0.028		ns/pF
SMC_{18}	NWR 高到片选停止 (5)	C_{NCS} = 0 pF	3.3		ns
		C_{NCS} 降额	0.028		ns/pF
SMC_{19}	在 NWR 高前数据出有效 (无等待状态) (1) (5)	C = 0 pF	t_{CHMCK} - 0.8		ns
		C_{DATA} 降额	- 0.044		ns/pF
		C_{NWR} 降额	0.045		ns/pF
SMC_{20}	在 NWR 高前数据出有效 (等待状态) (1) (5)	C = 0 pF	n x t_{CPMCK} - 0.6 (2)		ns
		C_{DATA} 降额	- 0.044		ns/pF
		C_{NWR} 降额	0.045		ns/pF
SMC_{21}	在 NWR 高后数据出有效 (无等待状态) (1) (5)	C = 0 pF	t_{CLMCK} - 1.0		ns
		C_{DATA} 降额	- 0.044		ns/pF
		C_{NWR} 降额	0.045		ns/pF
SMC_{22}	在 NWR 高后数据出有效 (无保持周期的等待状态) (1) (5)	C = 0 pF	t_{CHMCK} - 1.2		ns
		C_{DATA} 降额	- 0.044		ns/pF
		C_{NWR} 降额	0.045		ns/pF
SMC_{23}	在 NWR 高后数据出有效 (有保持周期的等待状态) (1) (5)	C = 0 pF	h x t_{CPMCK} - 1.1 (4)		ns
		C_{DATA} 降额	- 0.044		ns/pF
		C_{NWR} 降额	0.045		ns/pF



Table 121. SMC 写信号

符号	参数	条件	最小值	最大值	单位
SMC_{24}	在 NCS 高前数据出有效 (地址到片选建立周期) [1]	C = 0 pF	$(((n + 1) - nacss) \times t_{CPMCK}) + t_{CHMCK} - 1.4$ [2] [3]		ns
		C_{DATA} 降额	- 0.044		ns/pF
		C_{NCS} 降额	0.045		ns/pF
SMC_{25}	在 NCS 高后数据出有效 (地址到片选建立周期) [1]	C = 0 pF	$nacss \times t_{CPMCK} - 0.4$ [3]		ns
		C_{DATA} 降额	- 0.044		ns/pF
		C_{NCS} 降额	0.045		ns/pF
SMC_{26}	NWR 最小脉宽 (无等待状态) [1] [5]	C_{NWR} = 0 pF	$t_{CHMCK} - 0.1$		ns
		C_{NWR} 降额	0.002		ns/pF
SMC_{27}	NWR 最小脉宽 (等待状态) [1] [5]	C_{NWR} = 0 pF	$n \times t_{CPMCK}$ [2]		ns
		C_{NWR} 降额	0.002		ns/pF
SMC_{28}	NWR 最小脉宽 (地址到片选建立周期) [1]	C_{NWR} = 0 pF	$(n + 1) \times t_{CPMCK}$ [2]		ns
		C_{NWR} 降额	0.002		ns/pF

Notes:
1. 降额因子不适用于 t_{CLMCK}、 t_{CHMCK} 或 t_{CPMCK}。
2. n = 插入的标准等待状态数。
3. nacss = 插入的地址到片选建立周期数。
4. h = 插入的保持周期数。
5. 当地址到片选周期插入时不适用。

Table 122. SMC 读信号

符号	参数	条件	最小值	最大值	单位
SMC_{29}	MCK 下降到 NRD 激活 (1) (7)	C_{NRD} = 0 pF	4.5	6.8	ns
		C_{NRD} 降额	0.028	0.045	ns/pF
SMC_{30}	MCK 上升到 NRD 激活 (2)	C_{NRD} = 0 pF	4.7	7.0	ns
		C_{NRD} 降额	0.028	0.045	ns/pF
SMC_{31}	MCK 下降到 NRD 无效 (1) (7)	C_{NRD} = 0 pF	4.5	6.8	ns
		C_{NRD} 降额	0.028	0.045	ns/pF
SMC_{32}	MCK 下降到 NRD 无效 (2)	C_{NRD} = 0 pF	4.5	6.8	ns
		C_{NRD} 降额	0.028	0.045	ns/pF
SMC_{33}	在 MCK 下降前设置 D0-D15(8)		0.8		ns
SMC_{34}	在 MCK 下降后保持 D0-D15(9)		1.7		ns
SMC_{35}	NRD 高到 NUB 改变 (3)	C_{NUB} = 0 pF	(h x t_{CPMCK}) + 0.5 (6)	(h x t_{CPMCK}) + 0.8 (6)	ns
		C_{NUB} 降额	0.028	0.045	ns/pF
SMC_{36}	NRD 高到 NLB/A0 改变 (3)	C_{NLB} = 0 pF	(h x t_{CPMCK}) + 0.4 (6)	(h x t_{CPMCK}) + 0.7 (6)	ns
		C_{NLB} 降额	0.028	0.045	ns/pF
SMC_{37}	NRD 高到 A1-A25 改变 (3)	C_{ADD} = 0 pF	(h x t_{CPMCK}) + 0.3 (6)	(h x t_{CPMCK}) + 0.6 (6)	ns
		C_{ADD} 降额	0.028	0.045	ns/pF
SMC_{38}	NRD 高到片选无效 (3)	C_{NCS} = 0 pF	(h x t_{CPMCK}) - 0.3 (6)	(h x t_{CPMCK}) - 0.2 (6)	ns
		C_{NCS} 降额	- 0.045	- 0.028	ns/pF
SMC_{39}	片选无效到 NRD 高 (3)	C_{NCS} = 0 pF	(nacss x t_{CPMCK}) + 0.2 (5)	(nacss x t_{CPMCK}) + 0.3 (5)	ns
		C_{NCS} 降额	0.028	0.045	ns/pF
SMC_{40}	在 NRD 高前数据设置 (8)	C_{NRD} = 0 pF	7.5		ns
		C_{NRD} 降额	0.045		ns/pF
SMC_{41}	在 NRD 高后数据保持 (9)	C_{NRD} = 0 pF	-3.4		ns
		C_{NRD} 降额	- 0.028		ns/pF
SMC_{42}	在 NCS 高前数据设置	C_{NRD} = 0 pF	7.3		ns
		C_{NRD} 降额	0.045		ns/pF
SMC_{43}	NCS 高后数据保持	C_{NRD} = 0 pF	-3.2		ns
		C_{NRD} 降额	- 0.028		ns/pF
SMC_{44}	NRD 最小脉宽 (1) (3) (7)	C_{NRD} = 0 pF	n x t_{CPMCK} - 0.02 (4)		ns
		C_{NRD} 降额	0.002		ns/pF



Table 122. SMC 读信号

符号	参数	条件	最小值	最大值	单位
SMC_{45}	NRD 最小脉宽 (2) (3) (7)	C_{NRD} = 0 pF	n x t_{CHMCK} + t_{CHMCK} - 0.2 (4)		ns
		C_{NRD} 降额	0.002		ns/pF
SMC_{46}	NRD 最小脉宽 (2) (3)	C_{NRD} = 0 pF	((n + 1) x t_{CHMCK}) + t_{CHMCK} - 0.2 (4)		ns
		C_{NRD} 降额	0.002		ns/pF

Notes:
1. 初读协议。
2. 标准读协议。
3. 降额因子不适用于 t_{CHMCK} 或 t_{CPMCK}。
4. n = 插入的标准等待状态数。
5. nacss = 插入的地址到片选建立周期数。
6. h = 插入的保持周期数。
7. 当地址到片选周期插入时不适用。
8. 只需满足两个定时中的一个。
9. 只需满足两个定时中的一个。

Figure 270. 存储器接口模式中与 MCK 相关的 SMC 信号

Notes: 1. 初读协议
2. 有或没有设置与保持周期的标准读协议。



Figure 271. LCD 接口模式下与 MCK 相关的 SMC 信号

Notes: 1. 只有标准读协议。
2. 只插入标准等待状态。

与 SDCK 相关的 SDRAMC 信号

Table 123 与 Table 124 给出在 “Ã ºž”Î¹®þ±ºýÀ„” on page 593 给定的工作条件下的定时。

Table 123. SDRAMC 时钟信号

符号	参数	条件	最小值	最大值	单位
$1/(t_{CPSDCK})$	SDRAM 控制器时钟频率			80.0	MHz
t_{CPSDCK}	SDRAM 控制器时钟周期		12.5		ns
t_{CHSDCK}	SDRAM 控制器时钟高半周期		5.6		ns
t_{CLSDCK}	SDRAM 控制器时钟低半周期		6.9		ns

Table 124. SDRAMC 信号

符号	参数	条件	最小值	最大值	单位
$SDRAMC_1$	在 SDCK 上升沿前 SDCKE 高 (1)	$C_{SDCKE} = 0$ pF	$t_{CLMCK} + 1.2$		ns
		C_{SDCKE} 降额	0.015		ns/pF
$SDRAMC_2$	在 SDCK 上升沿后 SDCKE 低 (1)	$C_{SDCKE} = 0$ pF	$t_{CHMCK} - 1.4$		ns
		C_{SDCKE} 降额	- 0.023		ns/pF
$SDRAMC_3$	在 SDCK 上升沿前 SDCKE 低 (1)	$C_{SDCKE} = 0$ pF	$t_{CLMCK} + 1.0$		ns
		C_{SDCKE} 降额	0.015		ns/pF
$SDRAMC_4$	在 SDCK 上升沿后 SDCKE 高 (1)	$C_{SDCKE} = 0$ pF	$t_{CHMCK} - 1.7$		ns
		C_{SDCKE} 降额	- 0.023		ns/pF
$SDRAMC_5$	在 SDCK 上升沿前 SDCS 低 (1)	$C_{SDCS} = 0$ pF	$t_{CLMCK} + 1.2$		ns
		C_{SDCS} 降额	0.028		ns/pF
$SDRAMC_6$	在 SDCK 上升沿后 SDCS 高 (1)	$C_{SDCS} = 0$ pF	$t_{CHMCK} - 1.9$		ns
		C_{SDCS} 降额	- 0.045		ns/pF
$SDRAMC_7$	在 SDCK 上升沿前 RAS 低 (1)	$C_{RAS} = 0$ pF	$t_{CLMCK} + 0.6$		ns
		C_{RAS} 降额	0.015		ns/pF
$SDRAMC_8$	在 SDCK 上升沿后 RAS 高 (1)	$C_{RAS} = 0$ pF	$t_{CHMCK} - 1.1$		ns
		C_{RAS} 降额	- 0.023		ns/pF
$SDRAMC_9$	在 SDCK 上升沿前 SDA10 改变 (1)	$C_{SDA10} = 0$ pF	$t_{CLMCK} + 0.8$		ns
		C_{SDA10} 降额	0.015		ns/pF
$SDRAMC_{10}$	在 SDCK 上升沿后 SDA10 改变 (1)	$C_{SDA10} = 0$ pF	$t_{CHMCK} - 1.2$		ns
		C_{SDA10} 降额	- 0.023		ns/pF
$SDRAMC_{11}$	在 SDCK 上升沿前地址改变 (1)	$C_{ADD} = 0$ pF	$t_{CLMCK} + 0.6$		ns
		C_{ADD} 降额	0.028		ns/pF
$SDRAMC_{12}$	在 SDCK 上升沿后地址改变 (1)	$C_{ADD} = 0$ pF	$t_{CHMCK} - 1.5$		ns
		C_{ADD} 降额	- 0.045		ns/pF
$SDRAMC_{13}$	在 SDCK 上升沿前段改变 (1)	$C_{BA} = 0$ pF	$t_{CLMCK} + 0.8$		ns
		C_{BA} 降额	0.028		ns/pF

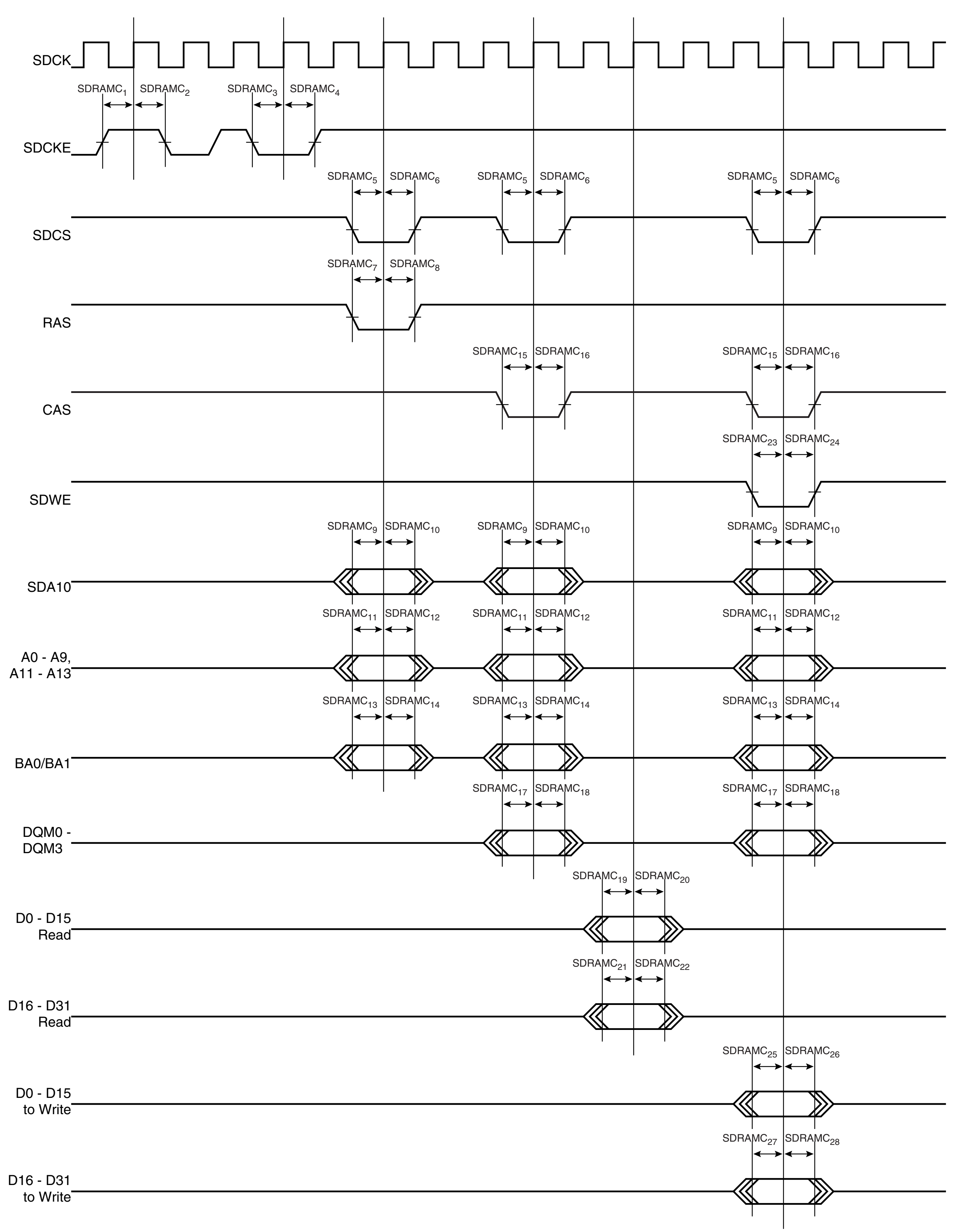


Table 124. SDRAMC 信号

符号	参数	条件	最小值	最大值	单位
$SDRAMC_{14}$	在 SDCK 上升沿后段改变 [1]	C_{BA} = 0 pF	t_{CHMCK} - 1.6		ns
		C_{BA} 降额	- 0.045		ns/pF
$SDRAMC_{15}$	在 SDCK 上升沿前 CAS 低 [1]	C_{CAS} = 0 pF	t_{CLMCK} + 0.9		ns
		C_{CAS} 降额	0.015		ns/pF
$SDRAMC_{16}$	在 SDCK 上升沿后 CAS 高 [1]	C_{CAS} = 0 pF	t_{CHMCK} - 1.5		ns
		C_{CAS} 降额	- 0.023		ns/pF
$SDRAMC_{17}$	在 SDCK 上升沿前 DQM 改变 [1]	C_{DQM} = 0 pF	t_{CLMCK} + 0.7		ns
		C_{DQM} 降额	0.028		ns/pF
$SDRAMC_{18}$	在 SDCK 上升沿后 DQM 改变 [1]	C_{DQM} = 0 pF	t_{CHMCK} - 1.4		ns
		C_{DQM} 降额	- 0.045		ns/pF
$SDRAMC_{19}$	在 SDCK 上升沿前 D0-D15 设置		1.3		ns
$SDRAMC_{20}$	在 SDCK 上升沿后 D0-D15 保持		0.03		ns
$SDRAMC_{21}$	在 SDCK 上升沿前 D16-D31 设置		2.0		ns
$SDRAMC_{22}$	在 SDCK 上升沿后 D16-D31 保持		-0.2		ns
$SDRAMC_{23}$	在 SDCK 上升沿前 SDWE 低	C_{SDWE} = 0 pF	t_{CLMCK} + 1.0		ns
		C_{SDWE} 降额	0.015		ns/pF
$SDRAMC_{24}$	在 SDCK 上升沿后 SDWE 高	C_{SDWE} = 0 pF	t_{CHMCK} - 1.8		ns
		C_{SDWE} 降额	-0.023		ns/pF
$SDRAMC_{25}$	在 SDCK 上升沿前 D0-D15 出有效	C = 0 pF	t_{CLMCK} - 2.7		ns
		C_{DATA} 降额	-0.044		ns/pF
$SDRAMC_{26}$	在 SDCK 上升沿后 D0-D15 出有效	C = 0 pF	t_{CHMCK} - 2.4		ns
		C_{DATA} 降额	-0.044		ns/pF
$SDRAMC_{27}$	在 SDCK 上升沿前 D16-D31 出有效	C = 0 pF	t_{CLMCK} - 3.2		ns
		C_{DATA} 降额	-0.044		ns/pF
$SDRAMC_{28}$	在 SDCK 上升沿后 D16-D31 出有效	C = 0 pF	t_{CHMCK} - 2.4		ns
		C_{DATA} 降额	-0.044		ns/pF

Note: 1. 降额因子不适用于 t_{CLMCK} 或 t_{CHMCK}。

Figure 272. 与 SDCK 相关的 SDRAMC 信号





与 BFCK 相关的 BFC 信号

Table 125、 Table 126 与 Table 127 给出在 “Ã º ž”Î¹®þ±ºýÀ„” on page 593 给定的工作条件下的定时。

Table 125. BFC 时钟信号

符号	参数	条件	最小值	最大值	单位
$1/(t_{CPBFCK})$	BF 控制器时钟频率	BFCK 为 MCK (1)		80.0	MHz
		BFCK 为 MCK/2 (2)		40.0	MHz
		BFCK 为 MCK/4 (3)		20.0	MHz
t_{CPBFCK}	BF 控制器时钟周期	BFCK 为 MCK (1)	12.5		ns
		BFCK 为 MCK/2 (2)	25.0		ns
		BFCK 为 MCK/4 (3)	50.0		ns
t_{CHBFCK}	BF 控制器时钟高半周期	BFCK 为 MCK (1)	6.5		ns
		BFCK 为 MCK/2 (2)	12.8		ns
		BFCK 为 MCK/4 (3)	25.3		ns
t_{CLBFCK}	BF 控制器时钟低半周期	BFCK 为 MCK (1)	6.1		ns
		BFCK 为 MCK/2 (2)	12.3		ns
		BFCK 为 MCK/4 (3)	24.8		ns

Notes: 1. 寄存器 BFC_MR 中域 BFCC = 1，见 “Burst Flash Controller Mode Register” on page 221。
2. 寄存器 BFC_MR 中域 BFCC = 2，见 “Burst Flash Controller Mode Register” on page 221。
3. 寄存器 BFC_MR 中域 BFCC = 3，见 “Burst Flash Controller Mode Register” on page 221。

Table 126. 异步模式下 BFC 信号

符号	参数	条件	最小值	最大值	单位
BFC_1	BFCK 上升到 A1-A25 有效 (1)	C_{ADD} = 0 pF		t_{CLBFCK} - 0.2	ns
		C_{ADD} 降额		- 0.028	ns/pF
BFC_2	BFCK 上升到 A1-A25 改变 (1)	C_{ADD} = 0 pF	t_{CLBFCK} - 1.0		ns
		C_{ADD} 降额	- 0.045		ns/pF
BFC_3	BFCK 下降到 BFAVD 激活 (1)	C_{BFAVD} = 0 pF	t_{CLBFCK} - 1.1	t_{CLBFCK} - 0.3	ns
		C_{BFAVD} 降额	- 0.044	- 0.028	ns/pF
BFC_4	BFCK 下降到 BFAVD 无效 (1)	C_{BFAVD} = 0 pF	t_{CLBFCK} - 1.8	t_{CLBFCK} + 0.2	ns
		C_{BFAVD} 降额	- 0.044	0.044	ns/pF
BFC_5	BFAVD 最小脉宽 (1)	C_{BFAVD} = 0 pF	t_{CPBFCK} + 1.0		ns
		C_{BFAVD} 降额	0.001		ns/pF
BFC_6	BFCK 上升到 BFOE 激活	C_{BFOE} = 0 pF	- 0.4	0.1	ns
		C_{BFOE} 降额	- 0.044	0.044	ns/pF
BFC_7	BFCK 上升到 BFOE 无效	C_{BFOE} = 0 pF	- 1.1	0.7	ns
		C_{BFOE} 降额	- 0.044	0.044	ns/pF

Table 126. 异步模式下 BFC 信号

符号	参数	条件	最小值	最大值	单位
BFC_8	BFOE 最小脉宽 (1)	C_{BFOE} = 0 pF	(a x t_{CPBFCK}) + 0.9 (2)		ns
		C_{BFOE} 降额	0.028		ns/pF
BFC_9	在 BFCK 上升沿前 D0-D15 设置 (5)		- 0.1		ns
BFC_{10}	在 BFCK 上升沿后 D0-D15 保持 (6)		1.0		ns
BFC_{11}	在 BFOE 高前数据设置 (5)	C_{BFOE} = 0 pF	- 0.9		ns
		C_{BFOE} 降额	- 0.044		ns/pF
BFC_{12}	在 BFOE 高后数据保持 (6)	C_{BFOE} = 0 pF	2.0		ns
		C_{BFOE} 降额	0.028		ns/pF
BFC_{13}	BFCK 上升到 BFWE 激活	C_{BFWE} = 0 pF	- 0.6	- 0.05	ns
		C_{BFWE} 降额	- 0.044	- 0.028	ns/pF
BFC_{14}	BFCK 上升到 BFWE 无效	C_{BFWE} = 0 pF	- 1.3	0.5	ns
		C_{BFWE} 降额	- 0.044	0.044	ns/pF
BFC_{15}	BFCK 上升到 AD0-AD15 有效 (1) (4)	C_{DATA} = 0 pF		t_{CLBFCK} - 0.2	ns
		C_{DATA} 降额		- 0.028	ns/pF
BFC_{16}	BFCK 上升到 AD0-AD15 无效 (1) (4)	C_{DATA} = 0 pF	t_{CLBFCK} - 0.8		ns
		C_{DATA} 降额	- 0.044		ns/pF
BFC_{17}	在 BFCK 上升前数据出有效 (1) (5)	C_{DATA} = 0 pF	t_{CLBFCK} + 0.5		ns
		C_{DATA} 降额	0.028		ns/pF
BFC_{18}	在 BFCK 上升后数据出有效 (1) (6)	C_{DATA} = 0 pF	t_{CHBFCK} + 0.7		ns
		C_{DATA} 降额	0.028		ns/pF
BFC_{19}	在 BFWE 高前数据出有效 (1) (5)	C = 0 pF	t_{CLBFCK} - 0.5		ns
		C_{DATA} 降额	- 0.028		ns/pF
		C_{BFWE} 降额	0.044		ns/pF
BFC_{20}	在 BFWE 高后数据出有效 (1) (6)	C = 0 pF	t_{CHBFCK} + 0.3		ns
		C_{DATA} 降额	0.028		ns/pF
		C_{BFWE} 降额	- 0.044		ns/pF
BFC_{21}	地址有效延迟周期数 (1)		((a + 1) x t_{CPBFCK}) (2)	((a + 1) x t_{CPBFCK}) (2)	ns
BFC_{22}	输出使能延迟周期数 (1)		(o x t_{CPBFCK}) (3)	(o x t_{CPBFCK}) (3)	ns

Notes: 1. 降额因子不适用于 t_{CPBFCK}。
2. a = 定义在 BFC_MR AVL 域的地址有效延迟周期数。
3. o = 定义在 BFC_MR OEL 域的输出使能延迟周期数。
4. 只可应用在复用地址及数据总线上。
5. 只需满足两个定时中的一个。
6. 只需满足两个定时中的一个。



Figure 273. 异步模式下与 BFCK 相关的 BFC 信号

Note: 1. 一旦 BFC_MR 寄存器中的 BFCOM 域不为 0， BFCS 出现。

Table 127. 脉冲模式下的 BFC 信号

符号	参数	条件	最小值	最大值	单位
BFC_1	BFCK 上升到 A1-A25 有效 [1]	C_{ADD} = 0 pF		t_{CLBFCK} - 0.2	ns
		C_{ADD} 降额		- 0.028	ns/pF
BFC_2	BFCK 上升到 A1-A25 改变 [1]	C_{ADD} = 0 pF	t_{CLBFCK} - 1.0		ns
		C_{ADD} 降额	- 0.045		ns/pF
BFC_3	BFCK 下降到 BFAVD 激活 [1]	C_{BFAVD} = 0 pF	t_{CLBFCK} - 1.1	t_{CLBFCK} - 0.3	ns
		C_{BFAVD} 降额	- 0.044	- 0.028	ns/pF
BFC_4	BFCK 下降到 BFAVD 无效 [1]	C_{BFAVD} = 0 pF	t_{CLBFCK} - 1.8	t_{CLBFCK} + 0.2	ns
		C_{BFAVD} 降额	- 0.044	0.044	ns/pF
BFC_5	BFAVD 最小脉宽 [1]	C_{BFAVD} = 0 pF	t_{CPBFCK} + 1.0		ns
		C_{BFAVD} 降额	0.001		ns/pF
BFC_6	BFCK 上升到 BFOE 激活	C_{BFOE} = 0 pF	- 0.4	0.1	ns
		C_{BFOE} 降额	- 0.044	0.044	ns/pF
BFC_7	BFCK 上升到 BFOE 无效	C_{BFOE} = 0 pF	- 1.1	0.7	ns
		C_{BFOE} 降额	- 0.044	0.044	ns/pF
BFC_9	在 BFCK 上升沿前 D0-D15 设置		- 0.1		ns
BFC_{10}	在 BFCK 上升沿后 D0-D15 保持		1.0		ns
BFC_{15}	BFCK 上升到 AD0-AD15 有效 [1] [4]	C_{DATA} = 0 pF		t_{CLBFCK} - 0.2	ns
		C_{DATA} 降额		- 0.028	ns/pF
BFC_{16}	BFCK 上升到 AD0-AD15 无效 [1] [4]	C_{DATA} = 0 pF	t_{CLBFCK} - 0.8		ns
		C_{DATA} 降额	- 0.044		ns/pF
BFC_{21}	地址有效延迟周期数 [1]		((a + 1) x t_{CPBFCK}) [2]	((a + 1) x t_{CPBFCK}) [2]	ns
BFC_{22}	输出使能延迟周期数 [1]		(o x t_{CPBFCK}) [3]	(o x t_{CPBFCK}) [3]	ns
BFC_{23}	BFCK 下降到 BFBAA 有效 [1]	C_{BFBAA} = 0 pF	t_{CLBFCK} - 1.0	t_{CLBFCK} - 0.1	ns
		C_{BFBAA} 降额	- 0.044	- 0.028	ns/pF
BFC_{24}	BFCK 下降到 BFBAA 无效 [1]	C_{BFBAA} = 0 pF	t_{CLBFCK} - 1.7	t_{CLBFCK} + 0.1	ns
		C_{BFBAA} 降额	- 0.044	0.044	ns/pF
BFC_{25}	在 BFCK 上升沿后 BFRDY 改变保持		0.1		ns
BFC_{26}	在 BFCK 上升沿前 BFRDY 改变设置		0.3		ns

Notes:
1. 降额因子不适用于 t_{CPBFCK}。
2. a = 定义在 BFC_MR AVL 域的地址有效延迟周期数
3. o = 定义在 BFC_MR OEL 域的输出使能延迟周期数。
4. 只可应用在复用地址及数据总线上。



Figure 274. 脉冲模式下与 BFCK 相关的 BFC 信号

BFCK if signal controlled address advance

BFBAA if signal controlled address advance

BFCK if clock controlled address advance

BFCS(1)

A1 - A25

BFAVD

BFRDY

BFOE

D0 - D15 Read

D0 - D15 to Write if multiplexed bus only

Note: 1. 一旦 BFC_MR 寄存器中的 BFCOM 域不为 0， BFCS 出现。

JTAG/ICE 定时

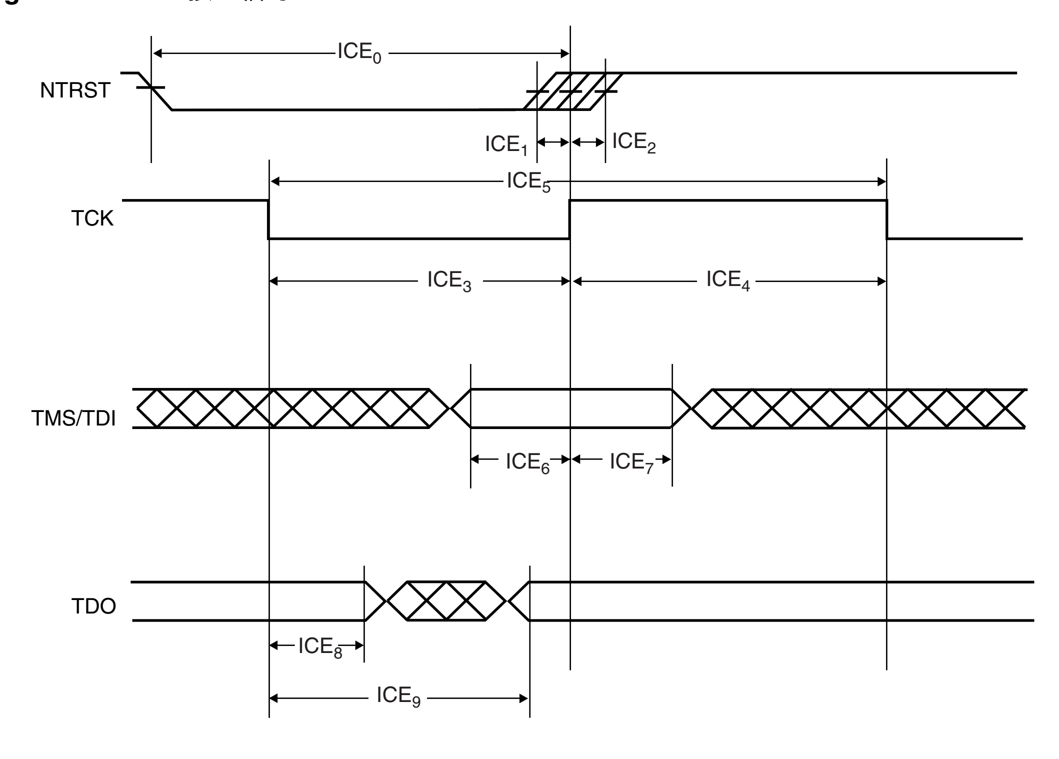
ICE 接口信号

Table 128 给出在 “Ã º̌ž”Î¹®þ±ºýÀ„” on page 593 给定的工作条件下的定时。

Table 128. ICE 接口定时规范

符号	参数	条件	最小值	最大值	单位
ICE_0	NTRST 最小脉宽		20.00		ns
ICE_1	NTRST 高恢复到 TCK 高		0.86		ns
ICE_2	TCK 高时 NTRST 高消除		0.90		ns
ICE_3	TCK 低半周期		8.00		ns
ICE_4	TCK 高半周期		8.00		ns
ICE_5	TCK 周期		20.00		ns
ICE_6	在 TCK 高前设置 TDI、 TMS		-0.13		ns
ICE_7	在 TCK 高后保持 TDI、 TMS		0.10		ns
ICE_8	TDO 保持时间	C_{TDO} = 0 pF	4.17		ns
		C_{TDO} 降额	0		ns/pF
ICE_9	TCK 低到 TDO 有效	C_{TDO} = 0 pF		6.49	ns
		C_{TDO} 降额		0.028	ns/pF

Figure 275. ICE 接口信号



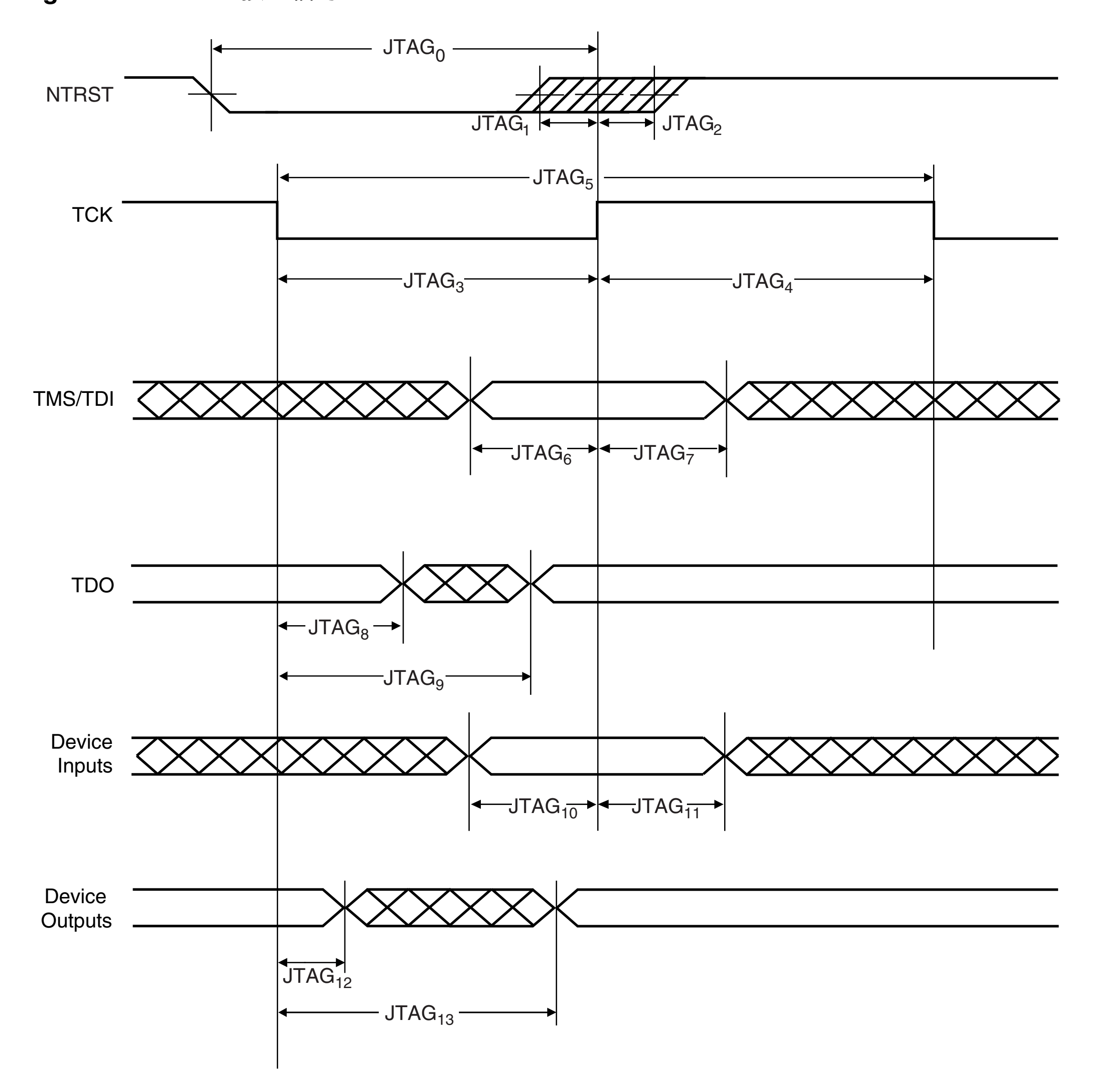
JTAG 接口信号

Table 129 给出在 "Ã ºž"Î¹®þ±ºýÀ„" on page 593 给定的工作条件下的定时。

Table 129. JTAG 接口定时规范

符号	参数	条件	最小值	最大值	单位
$JTAG_0$	NTRST 最小脉宽		20.00		ns
$JTAG_1$	NTRST 高恢复到 TCK 高		-0.16		ns
$JTAG_2$	NTRST 高恢复到 TCK 低		-0.16		ns
$JTAG_3$	TCK 高时 NTRST 高消除		-0.07		ns
$JTAG_4$	TCK 低时 NTRST 高消除		-0.07		ns
$JTAG_5$	TCK 低半周期		8.00		ns
$JTAG_6$	TCK 高半周期		8.00		ns
$JTAG_7$	TCK 周期		20.00		ns
$JTAG_8$	在 TCK 高前设置 TDI、 TMS		0.01		ns
$JTAG_9$	在 TCK 高后保持 TDI、 TMS		3.21		ns
$JTAG_{10}$	TDO 保持时间	C_{TDO} = 0 pF	2.38		ns
		C_{TDO} 降额	0		ns/pF
$JTAG_{11}$	TCK 低到 TDO 有效	C_{TDO} = 0 pF		4.66	ns
		C_{TDO} 降额		0.028	ns/pF
$JTAG_{12}$	器件输入设置时间		-1.23		ns
$JTAG_{13}$	器件输入保持时间		3.81		ns
$JTAG_{14}$	器件输出保持时间	C_{OUT} = 0 pF	7.15		ns
		C_{OUT} 降额	0		ns/pF
$JTAG_{15}$	TCK 到器件输出有效	C_{OUT} = 0 pF		7.22	ns
		C_{OUT} 降额		0.028	ns/pF

Figure 276. JTAG 接口信号



ETM 定时

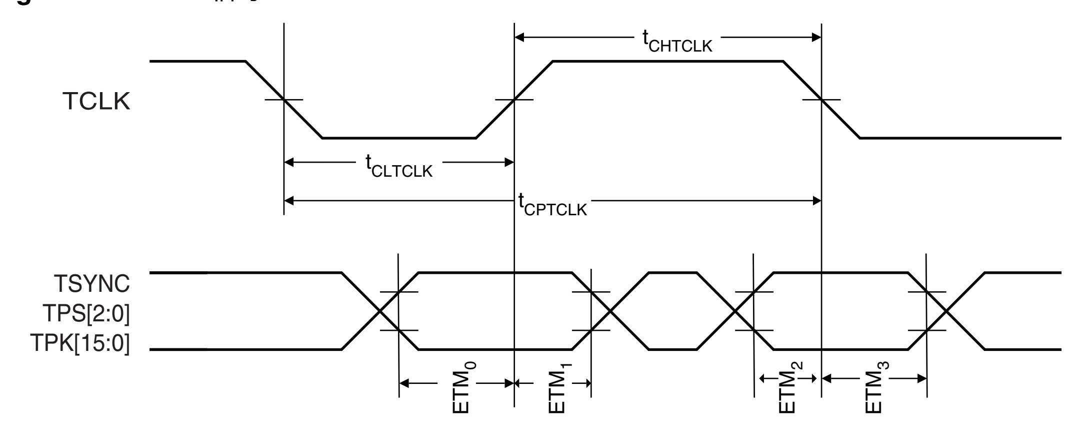
定时数据

Table 130 给出在 “Ã º ž”Î¹®þ±ºýÀ„” on page 593 给定的工作条件下的定时。

Table 130. ETM 定时特性

符号	参数	条件	最小值	典型值	最大值	单位
$1/(t_{CPTCLK})$	追踪时钟频率			$1/(2 \times t_{CPPCK})$	86.54	MHz
t_{CPTCLK}	追踪时钟周期		11.56	$2 \times t_{CPPCK}$		ns
t_{CHTCLK}	TCLK 高半周期		$t_{CPTCLK}/2 + 0.02$			ns
t_{CLTCLK}	TCLK 低半周期		$t_{CPTCLK}/2 - 0.02$			ns
ETM_0	在 TCLK 上升沿前数据信号出有效	C = 0 pF	$t_{CLTCLK} - 1.06$			ns
		C_{DATA} 降额	0.044			ns/pF
ETM_1	在 TCLK 上升沿后数据信号出有效	C = 0 pF	$t_{CHTCLK} - 0.49$			ns
		C_{DATA} 降额	0.044			ns/pF
ETM_2	在 TCLK 下降沿前数据信号出有效	C = 0 pF	$t_{CHTCLK} - 1.03$			ns
		C_{DATA} 降额	0.044			ns/pF
ETM_3	在 TCLK 下降沿后数据信号出有效	C = 0 pF	$t_{CLTCLK} - 0.51$			ns
		C_{DATA} 降额	0.044			ns/pF

Figure 277. ETM 信号



设计考虑

当设计 PCB 时，要保持 ETM 信号追踪长度的差别，以尽可能减小它们间的失真。此外，追踪端口串扰必须保持尽可能小，因为它会导致追踪结果错误。这些追踪抽头可引起无法预料的影响，因此建议避免在追踪线上出现抽头。

TCLK 线应为系列终端尽可能与微控制器引脚接近。

出现在追踪连接器、电缆及接口逻辑上的最大容阻必须低于 15 pF。

AT91RM9200 机械特性

温度及可靠性考虑

温度数据

Table 131 中，器件寿命估计使用“适度控制”环境模型的 MIL-217 标准 (该模型是指安置在一个有足够冷空气的固定架上)，依靠器件结温 (详见 “ 结温 ” on page 616)。

注意使用 MTBF 计算时要非常小心。注意根据 MIL-217 模型得到的值为保守值 (在严格的条件在测试)。寿命测试结果往往比预取的要好。

Table 131. MTBF 与结温的关系

结温 (T_J) (°C)	估计寿命 (MTBF) (年)
100	6
125	3
150	2
175	1

Table 132 给出各封装下的热阻。

Table 132. 热阻数据

<table>
<tr><th>符号</th><th>参数</th><th>条件</th><th>封装</th><th>类型</th><th>单位</th></tr>
<tr><td rowspan="2">θ_{JA}</td><td rowspan="2">环境结温热阻</td><td rowspan="2">静态空气</td><td>PQFP208</td><td>33.9</td><td rowspan="4">°C/W</td></tr>
<tr><td>LFBGA256</td><td>35.6</td></tr>
<tr><td rowspan="2">θ_{JC}</td><td rowspan="2">事件结温热阻</td><td rowspan="2"></td><td>PQFP208</td><td>15.7</td></tr>
<tr><td>LFBGA256</td><td>7.7</td></tr>
</table>

可靠性数据

门数及器件冲模尺寸见 Table 133，用户可在其它标准与 / 或其它环境模型下计算可靠性数据。

Table 133. 可靠性数据

参数	数据	单位
逻辑门数	4461	K 门
存储器门数	2458	K 门
器件冲模	33.9	mm^2

结温

平均芯片结温 T_J, 可按下式得到，单位 °C：

1. $T_J = T_A + (P_D \times \theta_{JA})$
2. $T_J = T_A + (P_D \times (\theta_{HEATSINK} + \theta_{JC}))$

其中：

- θ_{JA} = 封装热阻，环境结温 (°C/W) 见 Table 132 on page 615。
- θ_{JC} = 封装热阻，事件结温热阻 (°C/W) 见 Table 132 on page 615。
- $\theta_{HEAT\ SINK}$ = 制冷器件热阻 (°C/W)，由器件手册给出。
- P_D = 器件功耗 (W) 估计，来自 “Power Consumption” on page 598。
- T_A = 环境温度 (°C)。

根据第一个等式，可推导出芯片估计寿命，及决定是否需要制冷设备。若制冷设备不适合该芯片，使用第二个公式来计算平均芯片结温 T_J。

封装图

Figure 278. 208 引脚 PQFP 封装图

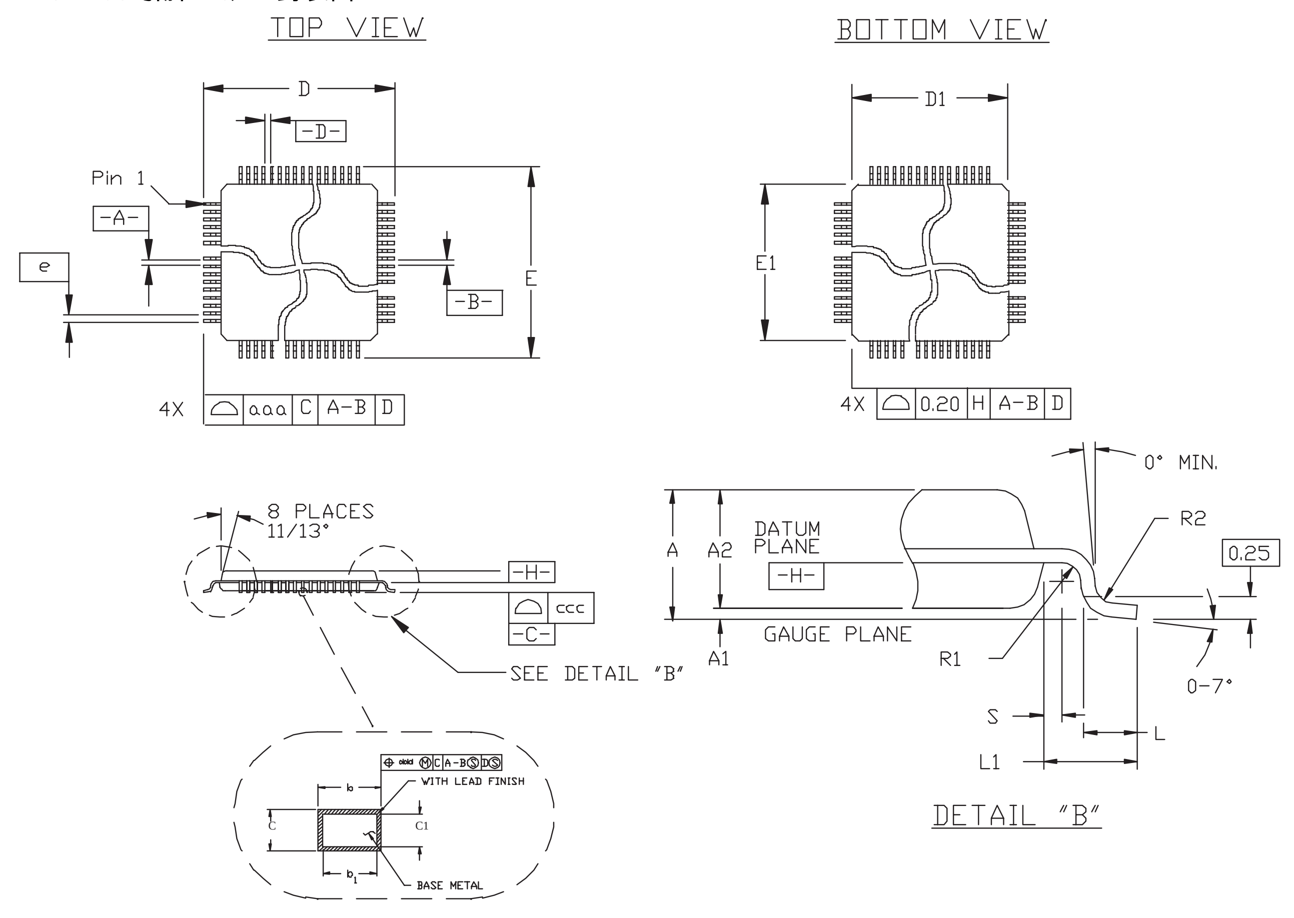
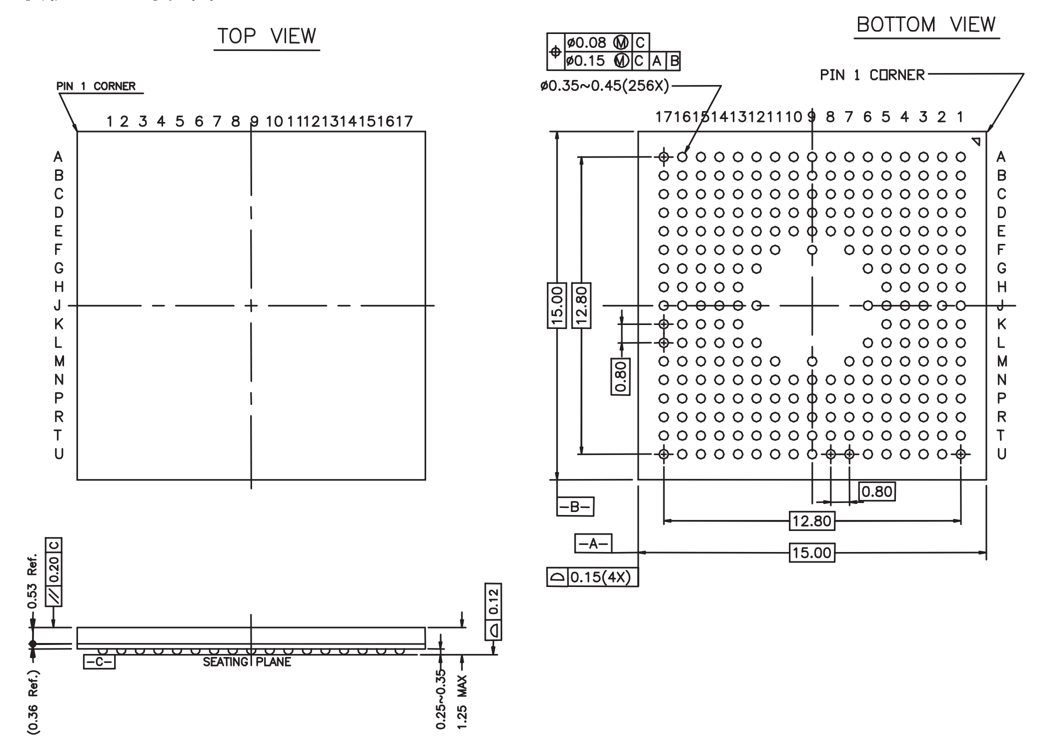


Table 134. 208 引脚 PQFP 封装尺寸 (单位：mm)

符号	最小值	名	最大值	符号	最小值	名	最大值
c	0.11		0.23	b1	0.17	0.20	0.23
c1	0.11	0.15	0.19	ddd	0.10		
L	0.65	0.88	1.03	形状与位置公差			
L1	1.60 REF			aaa		0.25	
R2	0.13		0.3	ccc			0.1
R1	0.13			**BSC**			
S	0.4			D	31.20		
A	4.10			D1	28.00		
A1	0.25		0.50	E	31.20		
A2	3.20	3.40	3.60	E1	28.00		
b	0.17		0.27	e	0.50		

Figure 279. 256 球形 BGA 封装图



AT91RM9200 订货信息

Table 135. 订货信息

订货号	封装	ROM 代码版本	温度 工作范围
AT91RM9200-QI-002	PQFP 208	002	工业级 (-40° C 到 85° C)
AT91RM9200-CI-002	BGA 256		

Table of Contents

































Atmel Corporation

2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

Regional Headquarters

Europe
Atmel Sarl
Route des Arsenaux 41
Case Postale 80
CH-1705 Fribourg
Switzerland
Tel: (41) 26-426-5555
Fax: (41) 26-426-5500

Asia
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
Fax: (852) 2722-1369

Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Atmel Operations

Memory
2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

Microcontrollers
2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

La Chantrerie
BP 70602
44306 Nantes Cedex 3, France
Tel: (33) 2-40-18-18-18
Fax: (33) 2-40-18-19-60

ASIC/ASSP/Smart Cards
Zone Industrielle
13106 Rousset Cedex, France
Tel: (33) 4-42-53-60-00
Fax: (33) 4-42-53-60-01

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

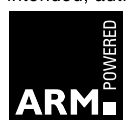
Scottish Enterprise Technology Park
Maxwell Building
East Kilbride G75 0QR, Scotland
Tel: (44) 1355-803-000
Fax: (44) 1355-242-743

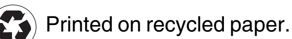
RF/Automotive
Theresienstrasse 2
Postfach 3535
74025 Heilbronn, Germany
Tel: (49) 71-31-67-0
Fax: (49) 71-31-67-2340

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Biometrics/Imaging/Hi-Rel MPU/
High Speed Converters/RF Datacom
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex, France
Tel: (33) 4-76-58-30-00
Fax: (33) 4-76-58-34-80

Literature Requests
www.atmel.com/literature





1768B–ATARM–07-Jun-05